Volatility Model Pricing and Calibration with Neural Networks using Bayesian Optimisation

Jenna Goosen

A dissertation submitted to the Faculty of Commerce, University of Cape Town, in partial fulfilment of the requirements for the degree of Master of Philosophy.

September 16, 2022

MPhil in Mathematical Finance,
University of Cape Town.

Declaration

I declare that this dissertation is my own, unaided work. It is being submitted for the Degree of Master of Philosophy at the University of the Cape Town. It has not been submitted before for any degree or examination in any other University.

September 16, 2022

Abstract

Stochastic Alpha, Beta, Rho (SABR) and Heston Volatility models have been used in the financial industry due to their ability to price options as a function of time to maturity and moneyness. Implied volatilities for these models are accurately estimated using a numerical integration approach, Monte Carlo approach, series-expansion approximation or using a two factor finite difference approach. However, these volatility calculations are computationally expensive. Therefore, this dissertation explores the use of deep artificial neural networks to calibrate volatility models by deploying computational resources to train a model (offline) and then utilise the pre-trained model to competitively price options (online). Deep neural networks in this paper are trained using an indirect and a direct method. The first step of the indirect method utilises a deep artificial neural network to approximate the pricing function (output) using the parameters as inputs. The second step of the indirect method implements a least squares optimisation algorithm to calibrate the parameters. The direct method, on the other hand, uses a deep artificial neural network to calibrate the model parameters using the implied volatility surface as input. Bayesian Optimisation algorithms are implemented to select models with the lowest loss metric for SABR and Heston volatility models. The best performing models are tested and compared for accuracy, speed and robustness for pricing and calibration. This dissertation finds that deep artificial neural networks using Bayesian Optimisation for the indirect and direct methods are able to efficiently and accurately price and calibrate the SABR and Heston Model. In addition, the results show that the implementation of the indirect and direct method, when there is no closed form approximation readily available, is advantageous from both a time and accuracy perspective for pricing and calibration.

Keywords: SABR, Heston, Deep Artificial Neural Networks, Calibration, Pricing Function, Bayesian Optimisation, At The Money Options

Acknowledgements

I would like to thank my supervisor, Prof. Peter Ouwehand who patiently provided me with guidance and support throughout the dissertation despite the difficult online working conditions and time pressure. I would also like to extend my thanks to the AIFMRM team for the teaching that enabled me to have the skills and knowledge for this dissertation.
I would like to acknowledge Joshua Cullinan for his continuous support, assistance and encouragement throughout the dissertation.
I would like to thank Raoul Goosen, Kelly Goosen and Joshua Cullinan for their valuable commentary when reviewing my dissertation
Finally, I would like to thank my family for their constant support and motivation. I am grateful for all the academic support that made the completion of this dissertation possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The Black-Scholes model has become the standard benchmark for option pricing in the financial industry. However, this model does not accurately capture the realities of financial markets (Hu (2013)). This dissertation looks at using stochastic volatility models to accurately and efficiently price options. The two volatility models explored in this paper are the Heston model and Stochastic Alpha, Beta, Rho (SABR) model. These models differ from the standard Black-Scholes in that they allow for the generalisation of implied volatility as a function of time to maturity and moneyness. Calibration of these models using market data can be extremely time consuming especially for volatility models that do not have explicit analytical solutions. Therefore, this paper explores the use of deep Neural Networks (NNs) to calibrate quickly and precisely whilst utilising large datasets.

The volatility models are calibrated by training deep NNs using an indirect and a direct method. The first step of the indirect method involves training the deep NN to predict an implied volatility surface (output) from the volatility model parameters (inputs) as a function of time to maturity and moneyness. The second step then uses the predicted implied volatility surface to calibrate model parameters using a weighted least squares optimisation algorithm. The direct method, on the other hand, uses a deep NN to predict the volatility model parameters (outputs) using the implied volatility (input) as a function of time to maturity and moneyness. Once the network is trained and the model parameters are calibrated under the direct and indirect method, the option is efficiently priced as a forward pass through the network. In addition, the pre-trained models are used to price at the money (ATM) options. This process is conducted for both the SABR and Heston volatility models. The aim of this dissertation focuses on the ability of Artificial Neural Networks (ANNs) to price using the indirect method and calibrate using the direct method. Calibration of the indirect method using weighted least squares

is conducted for completion purposes to investigate the ability of the two methods to then price options given calibrated parameters.

The dissertation is structured as follows: The remainder of Chapter 1 explores literature on volatility calibration, Chapter 2 provides an overview of the two models explored and their implied volatility smiles, Chapter 3 describes the methodology of model calibration and the optimisation procedure, Chapter 4 provides a discussion of results and model comparison and Chapter 5 concludes the dissertation.

The data, Bayesian Optimisation and best model NN code can be found on Github: https://github.com/JennaGoosen/Volatility-Model-Pricing-Calibration-with-Neural-Networks-using-Bayesian-Optimisation

1.2 Literature Review

From the inception of the Chicago Board Options Exchange, option trading has become the most common market activity amongst participants in the financial market, with more than 1 million contracts traded per day (Hu (2013)). The Black-Scholes model, being the standard benchmark model for call options, depends heavily on volatility. This was first recognised by Black (1976), who discovered an inverse relationship between stock prices and volatility. Since this realisation, intensive research has been completed in search for more accurate volatility measures to aid in identifying mispriced derivatives (Tripathy and Rahman (2013)). The ARCH model (introduced by Engle in 1982) used financial time series to model and forecast volatility (Hasanhodzic and Lo (2011)). These models were commonly used when time series data was found to have serial autocorrelation. Since then local volatility models have been explored. Dupire (1994) developed a local volatility model which incorporated the idea of replacing constant volatility in the Black-Scholes model with a function that is deterministic over a range of stock prices and time. However, local volatility models have since been proven to produce unrealistic volatility smiles and severely underestimate volatility over a period of time (Hagan *et al.* (2002)). This is due to the model incorporating prices of assets today in isolation, ignoring the behaviour of the prices over time. This has been proven to be unrealistic for various types of options and therefore led to research in the use of stochastic volatility modelling. Therefore, this dissertation will review two stochastic volatility models for pricing options, namely the SABR (Hagan *et al.* (2002)) and Heston (Heston (1993)) model.

Since the introduction of the the SABR and Heston volatility models there has been intensive research on calibration of these models using market data. Accurate

implied volatilities for these models can be produced using a numerical integration approach, Monte Carlo approach or using a two factor finite difference approach. However, these methods have many implementation challenges and are found to be non-performant when used on the SABR model (McGhee (2020)). Therefore, substantial research has been conducted to obtain more sophisticated representations of both the Heston and SABR models over a large parameter range and time domain (for example, Chan and Joshi (2010) for the Heston Model and McGhee (2020) for the SABR model). This led to the use of standard ANNs and deep NN for calibration to speed up the process (McGhee (2020) and Huang (2020)).

ANNs are comprised of an input layer (where the number of nodes equal to the number of inputs), one hidden layer and an output layer (where the number of nodes equal to number of outputs desired). Similarly, deep NNs are a type of ANNs that are comprised of an input and output layer. However, deep NNs make use of multiple hidden layers to construct complex relationships between the raw inputs and outputs. Figure 1.1 below illustrates the difference between ANNs and deep NNs.

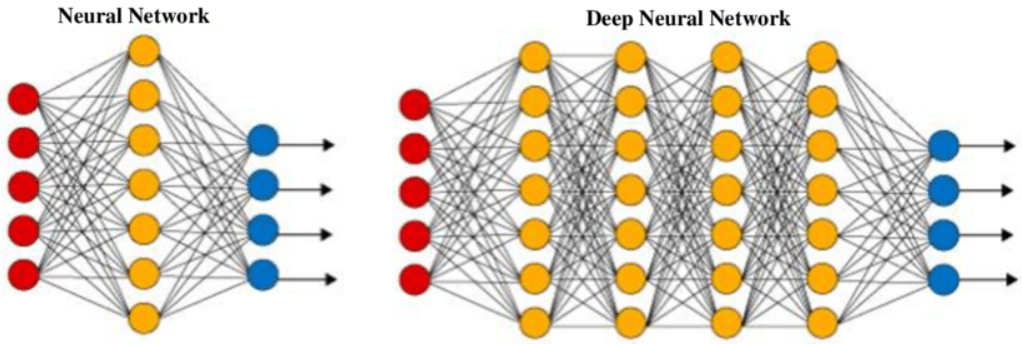

Fig. 1.1: NN and Deep Learning NN Comparison (Mostafa *et al.* (2020)).

Each layer consists of multiple learning units called nodes. The red nodes represent inputs, yellow nodes represent the hidden layers, and the blue nodes are the outputs (Figure 1.1). ANNs are typically constructed in such a way that each node within a layer is connected to all the subsequent layer's nodes and all the previous layer's nodes (except the bias node, discussed below). The value produced by each node is the sum of the previous layer's inputs (x_i) multiplied by the node's corresponding weights (w_i) plus a bias value. This value is then passed through an activation function which provides non-linearity to the NN. The activation function assumed in the equations below and used in this dissertation is the ReLU activation function. The nodes are activated when the sum of inputs of the individual nodes are found to be greater than the predetermined threshold. This is mathematically

expressed as follows:

$$\sum_i (w_i \times x_i) > \text{threshold}.$$

Alternatively, the equation above can be expressed in terms of bias:

$$\sum_i (w_i \times x_i) + \text{bias} > 0.$$

Weights are learnt in such a way that they control the strength of the signal between two nodes in the ANN. Therefore, nodes with larger weights in the input node will have a greater impact on the nodes in the next layer of the network. Conversely, biases (negative thresholds) are constant and are not influenced by the previous layer. Biases act as additional inputs to the next layer of the network and ensure that the node (in the next layer) will still be activated even if all input nodes are zero. Once the nodes are activated, data is sent to the next layer while passing through an activation function. The role of the activation function is to define the output node given an input or set of inputs. Activation functions are used to provide non-linearity to ANNs in order to accurately and realistically learn outputs. The ReLU activation function enables the ANN to learn outputs that vary non-linearly with the inputs. If a linear activation function were used to train a deep NN, the ANN would essentially only be one layer deep regardless of how many layers were incorporated into the network (i.e. producing similar results to simple linear regression). Therefore, this dissertation uses a non-linear activation function to create a non-linear mapping between inputs and outputs. The weights and biases are then continuously adjusted using a form of gradient descent. Gradient descent is an iterative algorithm that takes steps (determined by the size of the learning rate) in an opposite direction to the first derivative of the function. The learning rate is a hyperparameter in the ANN that requires tuning to ensure optimal weights and biases are selected. This algorithm is repeated until convergence (i.e. the minimum loss function is reached). This then produces a trained ANN with minimal difference between true outputs and predicted outputs.

The ability of an ANN to construct complex functional relationships is due to the Universal Approximation Theorem (Cybenko (1989)). This theorem states that using one hidden layer and activation function an ANN can approximate any function. However, when more than one hidden layer is included one can use fewer nodes which improves the efficiency of ANN. Therefore, more elaborate ANN configurations are used in order to infer complex relationships between the input and output layers. Examples of more intricate ANNs include memory networks, convolutional networks and recurrent networks (McGhee (2020)). However, research on volatility model calibration has proven that simple networks with very few hidden layers are able to encapsulate the dynamics of the financial model over a wide

range of parameter values and time domain (see for example McGhee (2020), Brigo *et al.* (2021) and Thorin (2020)).

ANNs have recently become a growing industry for their applications in the financial industry. The review article by Huang *et al.* (2020), explores the recent success of different deep learning models across the banking and finance sector. Rönnqvist and Sarlin (2017) trained deep learning models to predict levels of bank distress using text mining. This involved detecting relevant phrases in text and leveraging data to signal levels of bank distress. Another common use of deep learning models in finance is for forecasting. Sehgal and Pandey (2015) forecasts irregularities in oil prices, Shen *et al.* (2015) use deep belief networks and conjugate gradient methods to forecast exchange rate changes and Kim and Won (2018) uses artificial intelligence to predict volatilities for stock price indices. Huang *et al.* (2020) also explores the use of ANNs in finance from a portfolio management perspective (for example see Song *et al.* (2017)). Sevim *et al.* (2014) makes use of ANNs for macro economic prediction. This paper was able to predict the financial crisis in advance using a back propagation learning algorithm. Bennell and Sutcliffe (2004) applies daily and monthly data to demonstrate the use ANNs for option pricing and hedging. These papers are a few examples of uses of ANNs in a financial context.

It is clear from the recent papers mentioned above that there has been a sudden increase in interest in deep NNs which is credited to easily accessible high computational computing and big-data. Deep NNs require large amounts of data and are able to learn features and relationships from raw data offline and then use these relationships to predict results online. In connection to this, the improvement and ease of accessibility of ANN software and tools such as `PyTorch` (Paszke *et al.* (2019)) and `TensorFlow` (Abadi *et al.* (2016)) has enabled individuals with minimal coding skills to build complex ANNs. This has led to the increase of use of ANNs for financial model calibration.

A common method of model calibration, before the increase in support for ANNs, was a minimum distance algorithm (Liu *et al.* (2019)). This involved constructing an objective function that minimises the distance between the true and predicted observations for a given set of parameters. The advantage of this method was its transparency and simple mathematical implementation. Liu *et al.* (2019) used a parallel global optimisation method which provided fast and accurate results for stochastic volatility model calibrations. This technique prevented the optimisation algorithm getting stuck at local minimas as much as possible. Abu-Mostafa (2001) performed a slightly different approach which involved curve fitting based on the Kullback-Leibler distance error function and used consistency

hints to incorporate more information. Gupta (2010) implemented a Bayesian procedure for model calibration. This framework fitted an entire distribution for each calibrated parameters. However, these approaches were found to be unstable for certain initial parameter choices and computationally expensive for complex models. ANNs are able to provide a solution to these problems due to their ability to deploy computational resources in advance to produce a model offline. Pricing of the model then takes place online efficiently using the pre-trained model. This ensures that the model accurately captures the SABR and Heston model dynamics across a wide parameter space and over a time domain.

Volatility model calibration has been implemented in papers using both direct and indirect methods. Liu *et al.* (2019) experimented on calibration using the indirect method (two step algorithm for model parameter calibration) and found this framework able to calibrate high dimensional stochastic volatility models extremely accurately and efficiently. Similarly, Horvath *et al.* (2021) also implemented the indirect method which was able to calibrate rough volatility models for full volatility surfaces using deep NNs within milliseconds. This calibration took place both on simulated and historical data consisting of different derivative contracts. This paper also discussed the limitations of using Fourier pricing, PDE, Monte Carlo, etc. methods in comparison to the indirect method for pricing. The use of these methods was found to be substantially slower than when compared to the first step of the indirect method. In addition, selection of an optimal trade off between accuracy and efficiency was extremely difficult to achieve. Both Liu *et al.* (2019) and Horvath *et al.* (2021) found the indirect method to be highly accurate and computationally efficient. In connection, McGhee (2020) found the use of ANNs for SABR calibration to run 10,000 times faster than when compared to the finite difference scheme approach and 65 times faster when compared to the integration scheme approach. In addition to the substantial decrease in time, the ANN did not incur any of the weaknesses that were found when using the approximation of the SABR model. Hernandez (2016) used the direct method (mapping implied volatilities to model parameters directly) for the Hull and White model. Roeder and Dimitroff (2020) compared the indirect and direct method methods for the Heston and rough Bergomi model. This paper found the direct method's ability to calibrate model parameters better than when compared to model calibration using the indirect method. This dissertation will analyse the efficiency and accuracy of the SABR and Heston model under the direct and indirect methods.

Part of the process of training a successful ANNs comprises of good hyperparameter tuning. The advancement in research surrounding the topic of ANNs has also led to improvements in methods of hyperparameter tuning. This dissertation

will explore three common methods of hyperparameter tuning. The first method being manual parameter selection using both trial and error and recommendations from Huang (2020) and Thorin (2020). Another common method of parameter selection is using grid search. This involves looping through various combinations of parameters and monitoring which combination produces the lowest error. However, Bergstra and Bengio (2012) show that random hyperparameter selection using Bayesian Optimisation is more efficient for hyperparameter tuning than when compared to using a grid search method. Therefore, the last method of hyperparameter selection, which will be the main focus, is using Bayesian Optimisation.

The next chapter will provide an in depth explanation of the volatility models explored in this dissertation and the influence of their parameters on the implied volatility smile.

Chapter 2

Description of Volatility Models

In the Black-Scholes model there is a one to one relationship between the volatility parameter and the price of a European option (Black (1976)). Since this realisation, options have been quoted using their implied volatility rather than their price. However, as mentioned in Chapter 1, the Black-Scholes model is a constant volatility model. This led to the introduction of volatility models which allow volatilities to vary over a range of strikes and expiries. Handling market skews and smiles correctly has since become essential for accurate pricing of derivatives (Hagan *et al.* (2002)). Local volatility models were previously used to manage smile and skew risks. However, the dynamic behaviour of local volatility models incorrectly accounts for the relationship between the increase or decrease in price and its impact on smiles and skews. This led to the use of stochastic volatility models where the asset price and volatility are correlated. Therefore, this dissertation explores calibration of the SABR and Heston Model to price options.

2.1 SABR Model Dynamics

The SABR model is a stochastic model which is named after its parameters alpha, beta and rho (Hagan *et al.* (2002)). It is commonly used in derivative markets and captures the volatility surface (i.e. the evolution of volatility over time). In conjunction, the strike price and current forward price are also evolved over time.

The SABR Model is an asset process with correlated log-normal stochastic volatility (McGhee (2020)) and can be expressed using two factor dynamics. The first factor incorporates varying forward rate at time t whilst incorporating the second factor of random volatility. This is seen by the following Stochastic Differential Equations (SDEs):

$$\begin{aligned} dF_t &= \sigma_t (F_t)^{\beta} dW_t, \quad F_0 = f, \\ d\sigma_t &= \nu \sigma_t dZ_t, \quad \sigma_0 = \alpha, \\ dW_t dZ_t &= \rho dt. \end{aligned}$$

The initial value F_0 represents the discounted forward rate and σ_0 represents the initial volatility. W_t and Z_t are Wiener processes that exhibit the correlation between the underlying and its volatility. The parameter ρ dictates the slope of the skew, ν is the volatility of volatility, α controls the implied volatility level of ATM options and β controls the curvature of the model (McGhee (2020)).

Data is simulated for the SABR SDEs for implied volatility using the closed form approximation (with an accuracy of up to a series expansion). The closed form approximation for a discounted Forward (F) with a time to maturity (T) and strike (K) for $F \neq K$ is expressed as follows (Hagan *et al.* (2002)):

$$\sigma_{imp} = \frac{\alpha}{(FK)^{\frac{(1-\beta)}{2}}}\left[1+\left[\frac{(1-\beta^2)}{24}\frac{\alpha^2}{(FK)^{(1-\beta)}}+\frac{1}{4}\frac{\alpha\beta\nu\rho}{(FK)^{\frac{(1-\beta)}{2}}}+\frac{2-3\rho^2}{24}\nu^2\right]T\right] \times \frac{1}{1+\frac{1}{24}(1-\beta)^2(\ln(\frac{F}{K}))^2+\frac{1}{1920}(1-\beta)^4(\ln(FK))^4} \times \frac{\zeta}{D(\zeta)},$$

where:

$$\zeta = \frac{\nu}{\alpha}(FK)^{\frac{(1-\beta)}{2}}\ln\left(\frac{F}{K}\right),$$

$$D(\zeta) = \ln\left(\frac{\sqrt{1-2\rho\zeta+\zeta^2}+\zeta-\rho}{1-\rho}\right).$$

The closed form approximation for $F = K$ is found by taking the left hand limits for the expression $\frac{F}{K}$.

2.1.1 SABR Parameter Influences on Volatility Smile

The volatility smiles of the SABR parameters are plotted where ATM is satisfied when the strike is equal to 1. The influence of each of the parameters on implied volatility is observed. The model parameters in the figures below are arbitrarily chosen to be: $S = 1, r = 0,\ \beta = 1,\ \rho = 0.0419,\ \nu = 0.8,\ \alpha = 0.3936$ and $T = 0.1$.

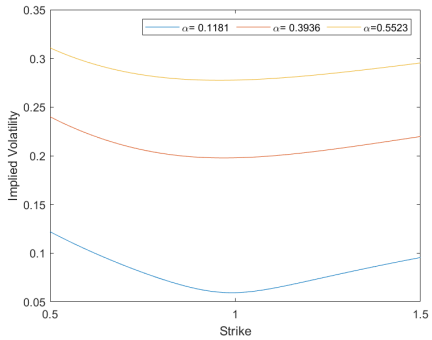

Fig. 2.1: α parameter influence.

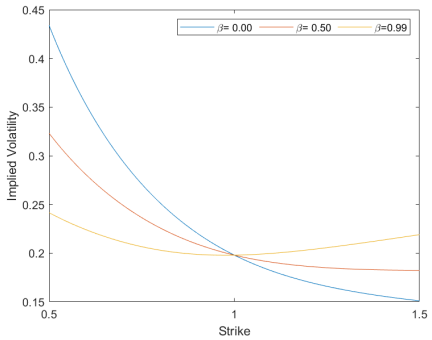

Fig. 2.2: β parameter influence.

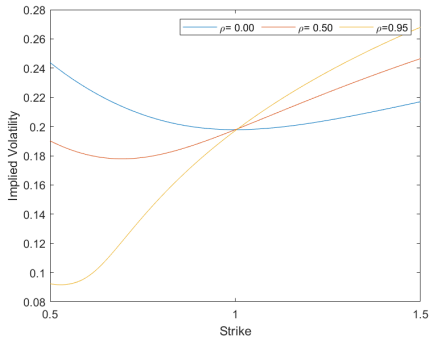

Fig. 2.3: Positive ρ parameter influence.

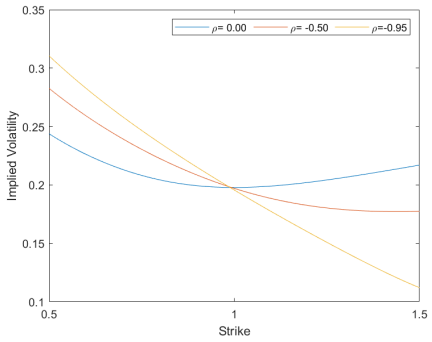

Fig. 2.4: Negative ρ parameter influence.

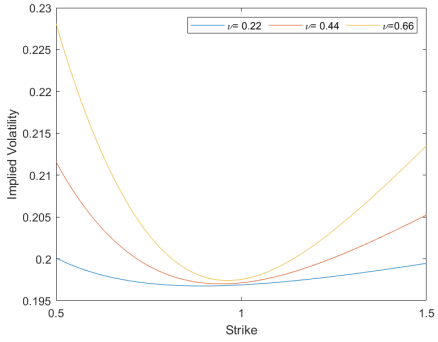

Fig. 2.5: ν parameter influence.

Figure 2.1 illustrates the higher the value of α the higher the curve is shifted. Figure 2.2 shows that the volatility smile differs for higher strikes. The lower the strike the steeper the smile. In addition, the higher the value of β the less steep the smile. This is inline with the dynamics of the SABR SDE where β is the power of the forward rate. Figure 2.3 and Figure 2.4 display the effects of positive and negative correlation between the volatility and the underlying. It is clear in the case of positive ρ, the higher the correlation the steeper the volatility smile. Conversely, negative ρ displays a steeper volatility smile as ρ decreases. Figure 2.2, Figure 2.3 and Figure 2.4 illustrate similar influences between β and ρ. Therefore, only ρ is used in model calibration. Lastly, Figure 2.5 influences the convexity of the curve with a lower value of ν resulting in a less convex volatility smile.

2.2 Heston Model Dynamics

Similarly to the SABR Model, the Heston model (Heston (1993)) is also used to price options whilst utilising the evolution of the underlying assets volatility. The Heston model is also expressed using two factor dynamics where both the underlying and the volatility of the underling follow a random process. The Heston model has five model parameters, namely $\nu_0, \kappa, \theta, \rho$ and σ. The Heston model is expressed using

the following two factor dynamics (assuming risk free interest rate (r) and dividend yield (q) are zero):

$$\begin{aligned} dS_t &= \sqrt{v_t} S_t dZ_t, \quad S_0, \\ d\nu_t &= \kappa(\theta - \nu_t)dt + \sigma\sqrt{\nu_t} dW_t, \quad \nu_0, \\ dW_t dZ_t &= \rho dt. \end{aligned}$$

The initial value S_0 represents the discounted asset price and ν_0 represents the initial volatility. W_t and Z_t are Wiener processes that represent correlation between the underlying and its volatility. The parameter ρ is used as a measure of this correlation, σ is the volatility of volatility, θ represents the long term variance of the asset and κ is the rate at which ν reverts to θ (Brigo *et al.* (2021)). The above model dynamics require that the Feller Condition ($2\kappa\theta > \sigma^2$) is satisfied in order for the process ν_t to be strictly positive.

Option prices are calculated using the Heston characteristic function, after which the implied volatility is found using the Black-Scholes equation. The European call price can be expressed under the risk neutral measure $\mathbf{Q}$ as follows (Thorin (2020)):

$$\begin{aligned} C(K) &= e^{-rT}\mathbf{E}^Q[(S_T - K)\mathbf{I}_{S_T > K}], \\ &= e^{-rT}\mathbf{E}^Q[S_T \mathbf{I}_{S_T > K}] - Ke^{-rT}\mathbf{E}^Q[\mathbf{I}_{S_T > K}], \end{aligned}$$

where r is the risk free interest rate, S_T is the stock price at time T and K is the strike price. The numeraire is changed by dividing through by the bank account ($\beta_t = e^{rt}$). The new measure $\mathbf{Q}_s$ is defined by $\frac{d\mathbf{Q}_s}{d\mathbf{Q}} = \frac{S_T/S_0}{\beta_T/\beta_0}$. The expectation of indicator functions can be expressed as probabilities under the new measure ($\mathbf{Q}_s$) and under the risk neutral measure ($\mathbf{Q}$) . Therefore, the price of a call option can be expressed as:

$$\begin{aligned} C(K) &= S_0 \mathbf{Q_s}\{S_T > K\} - Ke^{-rT}\mathbf{Q}\{S_T > K\}, \\ &= S_0 \mathbf{Q_s}\{\ln(S_T) > \ln(K)\} - Ke^{-rT}\mathbf{Q}\{\ln(S_T) > \ln(K)\}, \\ &= S_0 P_1 - Ke^{-rT} P_2 \end{aligned}$$

P_1 and P_2 in the Black-Scholes context are $\Phi(d_1)$ and $\Phi(d_2)$. Under the Heston model, these probabilities are computed using a suitable characteristic function whilst following the Gil-Pelaez theorem.

Theorem 2.1: Gil-Pelaez (Gil-Pelaez (1951))
Let ϕ_X be the characteristic function associated with random variable X, then, for $k \in \mathbf{R}$,

$$Pr\{X > k\} = \frac{1}{2} + \frac{1}{\pi}\int_0^\infty Re\left(\frac{e^{-iuk}\phi_X(u)}{iu}\right) du.$$

Therefore setting $s_T = \ln(S_T)$ and $k = \ln(K)$, the probability P_2 can be computed as (Zhylyevskyy (2012)):

$$\mathbf{Q}\{s_T > k\} = \frac{1}{2} + \frac{1}{\pi}\int_0^\infty Re\left(\frac{e^{-iuk}\phi_X(u)}{iu}\right) du.$$

P_1 is computed using the change of measure (Zhylyevskyy (2012)):

$$\frac{d\mathbf{Q}_s}{d\mathbf{Q}} = \frac{S_T}{\mathbf{E}^{\mathbf{Q}}[S_T]} = \frac{e^{s_T}}{\mathbf{E}^{\mathbf{Q}}[e^{s_T}]} = \frac{e^{s_T}}{\phi_{s_T}(-i)}.$$

The change of measure allows one to relate the density of the model under the new measure to the density under the risk neutral measure. The relationship between the characteristic function of s_T and under Q_s is given by (Albrecher *et al.* (2007)):

$$\phi^s_{s_T}(u) = \frac{\phi_{s_T}(u-i)}{\phi_{s_T}(-i)}.$$

Therefore, P_1 is computed as:

$$\mathbf{Q}_s\{s_T > k\} = \frac{1}{2} + \frac{1}{\pi}\int_0^\infty Re\left(\frac{e^{-iuk}\phi_X(u-i)}{iu\phi_{s_T}(-i)}\right)du.$$

The characteristic function is computed using the Little Heston Trap formulation (Albrecher *et al.* (2007)). The Little Heston Trap formulation is found to be numerically more stable when compared to Heston's original derivation of the Heston characteristic function. Therefore, the European call option is priced using the Little Heston Trap formulation of the characteristic function as follows:

$$\phi_{s_T}(u) = e^{C+D\nu_0+iu\ln(S_0)},$$

where:

$$\begin{aligned} C =& rTiu + \theta\kappa\left(Tx_- - \frac{1}{a}\ln\left(\frac{1-ge^{-Td}}{1-g}\right)\right), \\ D =& \frac{1-e^{-Td}}{1-ge^{-Td}}x_-, \end{aligned}$$

with:

$$\begin{aligned} a =& 0.5\sigma^2, \quad b = \kappa - \rho\sigma iu, \quad c = -\frac{u^2+iu}{2}, \\ d =& \sqrt{b^2-4ac}, \quad x_\pm = \frac{b\pm d}{2a}, \quad g = \frac{x_-}{x_+}. \end{aligned}$$

Data is simulated using the characteristic pricing method with 100 quadrature steps and an upper integration limit of 30 (i.e $u_{max} = 30$). Therefore, the integrals derived using Gil-Pelaez theorem are truncated as follows:

$$P_1 = \frac{1}{2} + \frac{1}{\pi}\sum_{n=1}^{N} Re\left(\frac{e^{-iu_nk}\phi_X(u_n-i)}{iu_n\phi_{s_T}(-i)}\right)\Delta u,$$

$$P_2 = \frac{1}{2} + \frac{1}{\pi}\sum_{n=1}^{N} Re\left(\frac{e^{-iu_nk}\phi_X(u_n)}{iu_n}\right)\Delta u,$$

Where $\Delta u = \frac{u_{max}}{N}$ and $u_n = (n-\frac{1}{2})\Delta u$.

Then the implied volatility is obtained by setting the option price using the characteristic function equal to the Black-Scholes option price over a range of time to maturity and level of moneyness.

2.2.1 Heston Parameter Influences on Volatility Smile

The following figures show the influence of changing each parameter on the volatility smile of the Heston Model. The model parameters not being varied in the figures below are arbitrarily chosen as: $S = 1, r = 0,\ \theta = 0.05,\ \kappa = 9,\ \nu = 0.05,\ \rho = 0.00$ and $T = 0.75$.

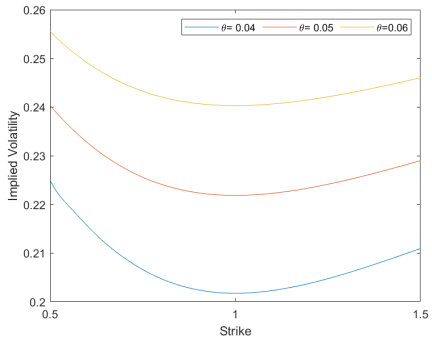

Fig. 2.6: θ parameter influence.

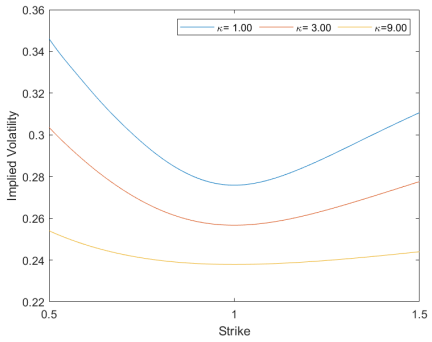

Fig. 2.7: κ parameter influence.

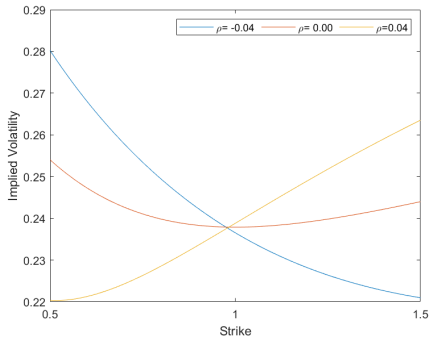

Fig. 2.8: ρ parameter influence.

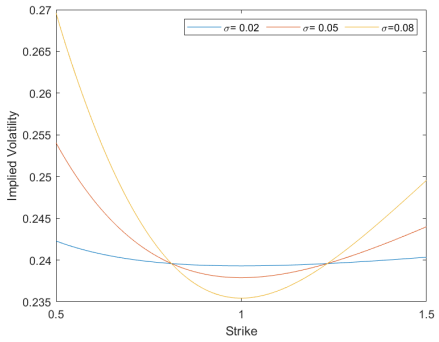

Fig. 2.9: σ parameter influence.

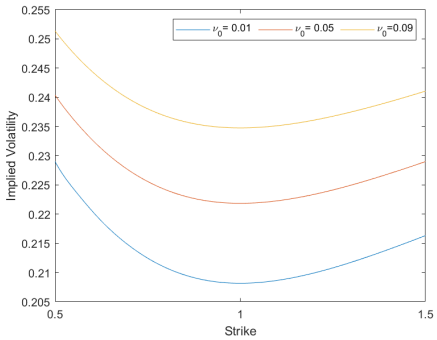

Fig. 2.10: ν_0 parameter influence.

Figure 2.6 illustrates the higher the value of θ the higher the curve. Figure 2.7 influences the convexity of the curve with a lower value of κ resulting in a more convex volatility smile. Figure 2.8 illustrates the slope of the volatility smile, where negative correlations display a downward volatility skew and positive correlations display an upward volatility skew. Figure 2.9 displays a combination of the effects seen from parameters κ and θ. σ effects both the convexity and the shift of the volatility smile. Lastly, Figure 2.10 has a similar influence to parameter θ and shifts the curve. Higher values of ν_0 result in an upward shift of the volatility smile.

Chapter 3

Model Calibration Methodology

3.1 Data Generation & Parameter Bound Selection

This section explains how synthetic datasets were simulated for both the SABR and Heston models in order to build the training and validation datasets for model calibration. Before data generation occurred, parameter bounds were selected for both the SABR and Heston models.

3.1.1 Parameter Bounds

SABR Model

The SABR model has the following parameters: α, β, ν and ρ. However, as mentioned in Chapter 2, β and ρ have similar influences. Therefore, only ρ is used in model calibration and $\beta = 1$ using traders' intuition (Thorin (2020)).

Since the closed form approximation of the SABR model is used for data generation, arbitrary parameters bounds are selected for values under which the closed form approximation is numerically stable: Table 3.1 defines the lower and upper bounds of each parameter.

Parameter	α	ν	ρ
Lower Bound	0.00	0.00	-0.95
Upper Bound	1.00	1.00	0.95

Tab. 3.1: SABR Model Parameter Bounds.

Heston Model

The Heston model has the following parameters: $\nu_0, \kappa, \theta, \rho$ and σ. Model calibration for this model occurs for all five parameters. The parameter bounds selected are displayed in Table 3.2 below:

Parameter	ν_0	θ	κ	σ	ρ
Lower Bound	0.01	0.04	1	0.1	-0.4
Upper Bound	0.08	0.06	10	0.8	0.4

Tab. 3.2: Heston Model Parameter Bounds.

Once the parameters were obtained using the bounds stated above, the data was then refined to satisfy the Feller Condition as stated in Chapter 2.

3.2 Data Generation

Model calibration for each model is conducted following the indirect and direct method. The first step of the indirect method uses a deep NN to approximate the pricing function (implied volatility surface) as outputs using the parameters as inputs. After which, a weighted least squares optimisation is performed using the deep NN predicted pricing function to calibrate model parameters. Conversely to the first step of the indirect method, the direct method uses a deep NN to approximate the parameters as outputs using the implied volatility surface as inputs. The aim of indirect method is to obtain a faster version of the pricing map without learning the whole calibration procedure. The aim of the direct method is to obtain a calibration procedure that is substantially faster than compared to traditional least squares optimisation calibration approaches. However, for the SABR model, there is a closed form solution (Hagan *et al.* (2002)) which is extremely fast to compute, even faster than a well performing simple ANN. Therefore, the aim of using the indirect and direct method for the SABR volatility model is to experiment whether ANNs are able to calibrate and price accurately. This is used as a foundation for calibration and pricing of the more complex and memory intensive Heston model.

The aim of data generation under both methods is to randomly simulate a large amount of data under a wide selection of parameter ranges, levels of moneyness and time domain.

Implied volatilities for the SABR and Heston model are generated as explained in Chapter 2. SABR implied volatilities are generated using the closed form approximation whilst the characteristic pricing function is used for the Heston model and implied volatilities are then found using the Black-Scholes equation.

The steps of data generation for the SABR and Heston model under the indirect and direct method are as follows:

- One of each parameter is randomly generated using a uniform distribution within the bounds specified in Table 3.1 and Table 3.2.
- A grid of maturities (T) and strikes (K) is generated as follows:

$$K = \{0.6, 0.8, 1, 1.2, 1.4, 1.6\},$$

$$T = \{0.50, 0.75, 1.00, 1.25, 1.50, 1.75\}.$$

- Fixed variables:
  - For the Heston model: set $S_0 = 1$ and $r = 0$.
  - For the SABR model: set $f = 1$, $\beta = 1$ and $r = 0$.
- Implied volatilities are calculated (as explained in Chapter 2) for each value of K, T and the randomly generated parameter values. This creates a volatility matrix of 6 strikes per 6 maturities.
- The implied volatility matrix is flattened which results in a feature selection of $6 \times 6 = 36$.
- The values of each parameter and implied volatilities are concatenated into a row vector.
  For example:
  - Heston model has 5 parameters and 36 implied volatilities. Hence, a row vector 1×41 is created. Similarly, the SABR model has 3 parameters and 36 implied volatilities. Hence, a row vector of 1×39.
- The above steps are repeated $100\,000$ times to simulate $100\,000$ rows of data.

For this dissertation the data generation method for each model was implemented using `Matlab 2020b`. The data was then stored and imported into `Python 3.9.7` for model calibration.

3.3 Model Optimisation Problem

Calibration of model parameters is achieved by minimising the square errors between the outputs obtained and the targets required. The minimisation problem in this dissertation was set up using the two methods.

- Indirect Pricing Method:
  The indirect pricing method explores minimising the error of the simulated true implied volatilities and the predicted implied volatilities over a grid of strikes (**K**) and times to maturity (**T**). This is achieved by using a deep NN to find optimal values for $\hat{\sigma}_{imp}^{pred}$.

  $$\hat{\sigma}_{imp}^{pred} = \arg\min_{\sigma_{imp}^{pred}} \sum_i (\sigma_{imp_i}^{sim} - \sigma_{imp}^{pred}(\mathbf{K}, \mathbf{T}, \theta_i))^2,$$

  where $\sigma_{imp_i}^{sim}$ represents each set of 36 simulated implied volatilities and θ_i represents the parameters used to simulate each set of 36 predicted implied volatilities ($\sigma_{imp_i}^{sim}$). For example, $\theta = \{\nu_0, \theta, \kappa, \rho, \sigma\}$ for the Heston Model and $\theta = \{\nu, \alpha, \rho\}$ for the SABR model (since $\beta = 1$).

- Indirect and Direct Calibration Method:
  Similarly, the indirect and direct calibration method aims to minimise the error of the simulated true implied volatilities and the predicted implied volatilities over a grid of strikes ($\mathbf{K}$) and times to maturity ($\mathbf{T}$). However, this is achieved by obtaining optimal parameters values for θ.

$$\hat{\theta} = \arg\min_{\theta} \sum_{i} (\sigma_{imp_i}^{sim} - \sigma_{imp}^{pred}(\mathbf{K}, \mathbf{T}, \theta))^2,$$

  where $\sigma_{imp_i}^{sim}$ represents the set of 36 simulated implied volatilities and $\hat{\sigma}_{imp_i}^{pred}$ represents the set of 36 predicted implied volatilities using optimal parameters for θ. The optimal values for θ under the indirect method are obtained by using a weighted least squares algorithm whilst the optimal values for the direct method are found using a deep NN.

3.4 Software and Hardware

The deep NNs in this dissertation were trained using `Python 3.9.7` and `TensorFlow 2.6.0`. Calculations were primarily run using a machine with an Intel Core i5-6300 CPU @ 2.40GHz with 2 cores and 4 logical processors and 8GB of RAM. For the more memory intensive models a second machine was used with an AMD Ryzen 5 3600 CPU @ 4.00 Ghz with 6 cores and 12 logical processors with an NVIDIA GeForce RTX 3070 GPU and 32GB of RAM.

3.5 Model Architecture

Correct model architecture choice is essential for good model calibration performance. The input layer and output layer number of units is determined by the number of inputs and outputs provided. For example, when approximating implied volatility (output) using the model parameters as inputs for the Heston Model one would have five input nodes ($\nu_0, \theta, \kappa, \rho, \sigma$) and thirty six output nodes ($\sigma_{imp}{}^{pred}$) In addition to the input and output layer, the following model dynamics are also determined:

- Training, Validation and Test Set Split:
  Data generation involved simulating $100\,000$ data points for each model and approach. $30\,000$ of these data points were allocated to the test set and the remaining $70\,000$ were split (0.9 and 0.1) between the training and validation dataset.

- Activation Function:
  The ReLU activation function (defined by $\text{ReLU}(x) = \max(x, 0)$) is used for hidden layers and linear for the output layer. One of the advantages of using the ReLU activation function is that not all neurons are activated at the same time. Negative input values result in the neurons being deactivated. This makes the ReLU activation function much more efficient than when compared to other non-linear activation functions. The linear activation function

(also known as "no activation") is used for the output because it returns the value directly without changing the weighted sum of the inputs. Therefore, the linear activation function is used for the output layer.

- Loss Function:
  Since this is a regression problem, a regression loss function is needed. Both the Mean Absolute loss function ($\frac{1}{N}\sum_{i=0}^{N}|y-\hat{y}_i|$) and Mean Square error (MSE) loss function ($\frac{1}{N}\sum_{i=0}^{N}(y-\hat{y}_i)^2$) were tested. However, there was no significant change in outputs. Therefore, either loss function is suitable. This paper uses the MSE loss function. A standard MSE loss function is used for the indirect pricing ANN. However, a customised MSE loss function is used for direct method. This customisation involves weighting each loss by the scale of the outputs and the performance of each model calibration separately. For example, in the Heston model, parameters ν_0 and θ have a much smaller range than when compared to κ. Therefore, the loss is scaled by scaling up the weighting of ν_0 and θ on the total loss and scaling down the weighting of κ on the total loss. The weightings selected for the Heston model and SABR Model under the direct method are as follows:
  - SABR Model: [0.25, 0.25, 0.5] for the parameters ν, α and ρ.
  - Heston Model: [0.0191, 8.57e-05,0.7626, 8.57e-03, 0.0381] for the parameters ν_0, κ, θ, ρ and σ.
- Optimiser:
  The Optimiser choice is Adam (Kingma and Ba (2014)). Adam incorporates momentum which helps make gradient descent smoother. In addition, Adam requires low memory and is efficient for small change in hyperparameters.
- Early Stopping:
  Early stopping criteria are used in ANN to stop training when there is no significant improvement in the validation dataset. This ensures that the model is able train long enough to learn the complex relationships between inputs and outputs but to stop before overfitting the training data. This dissertation stops training if the loss metric has not improved by 0.000001 after 40 epochs. Each epoch includes a complete pass of the entire training dataset through the model.
- Learning Rate Scheduler:
  A learning rate scheduler is used to adjust the learning rate during training when the loss metric is not improving for the validation set. The learning rate is reduced by a factor of 10% if there is no improvement in the loss function for more then 15 epochs.

The remaining architecture of the ANN requires very precise tuning of parameters and model hyperparameters. This dissertation will explore tuning of the following hyperparameters:

- Batch Size:
  Batch size refers to a portion that is sampled from the dataset and is propagated through the ANN. For example, consider a training dataset with $35\,000$ datapoints and a batch size of 100. The ANN will take the first 100 (1-100) samples of the training data to train the network and then will move onto the next 100 (101-200) samples and so on until all the training data is utilised.

- Number of Epochs:
  This refers to the number of times the entire training dataset is passed through the ANN.

- Number of Layers:
  ANNs always consists of input and output layers, however, the depth of the network is varied. The number of layers refers to the number of hidden layers in the ANN.

- Number of Nodes:
  This hyperparameters refers to the number of nodes within each hidden layer.

- Learning Rate:
  The learning rate determines the size of the step taken towards the minimum loss. The higher the value of the learning rate the higher the risk of overshooting the minimum loss. However, the lower the learning rate the longer the training and the possibility of predicting sub-optimal weights increases. Therefore, the learning rate hyperparameter chosen is required to find a fine balance between overshooting and getting stuck on sub-optimal weights.

- Presence of Dropout:
  Dropout is a technique used in ANN to prevent the model from overfitting. This is where nodes in each layer are randomly selected and ignored during training.

3.6 Hyperparameter Tuning

The aim of hyperparameter optimisation is to find a certain combination of parameters which result in an ANN architecture that performs best on the validation set (i.e. minimises the validation loss metric). However, finding these hyperparameters is computationally expensive and may, in the case of training a deep NN, take many days to train. This process involves training the model on training set and minimising the loss of the validation set. The process is then repeated for numerous combinations of hyperparameters and models performance metrics are logged to find the model with the best loss. This dissertation explored three types of hyperparameter selection:

- Manual Parameter Selection:
  This involves hand tuning the hyperparameters and observing the improvement in the loss metric. Recommendations of optimal hyperparameters from

Huang (2020) and Thorin (2020) were also used to improve the manual parameter selection.

- Grid Search:
  This involves looping through various combinations of pre-specified parameters. The model is then trained, predicted and evaluated for one combination and then automatically run to train the next combination. This was run overnight for both the SABR and Heston model (under the indirect pricing and direct calibration method) and the combination of parameters that produced the lowest error was saved.

- Bayesian Optimisation:
  Bayesian Optimisation is an extension of random search. Random search follows a similar method to grid search with the exception that parameters values are randomly calculated rather than being pre-specified. Bayesian Optimisation differs from random search since past results are used to restrict the range of hyperparameters to a smaller region which is more likely to produce the optimal model. Bayesian Optimisation keeps track of previous evaluation results and strategically updates the hyperparameters using prior knowledge of the previous hyperparameters performance. The aim of Bayesian Optimisation mathematically is to minimise the following expression:

  $$x^* = \text{argmin}_x \, f(x),$$

  where x represent the hyperparameters that require tuning and $f(x)$ represents the objective (MSE) function. The Bayesian Optimisation technique is well suited in the case of ANN hyperparameter tuning. This is because $x^* = \text{argmin}_x \, f(x)$ has the following constraints: Firstly f is computationally expensive to evaluate. Secondly, f is a black box function with no closed form solution and lastly, many observations of $f(x)$ are noisy. These properties are commonly associated with stochastic volatility models and benefit from the use of ANNs for prediction. Therefore, this has resulted in Bayesian Optimisation being commonly used for various types of Machine learning. Joy *et al.* (2016) uses Bayesian optimisation to generate hyperparameter configurations for chunks of data using deep NNs. This paper found Bayesian Optimisation to produce the best available configuration of hyperparameter more efficiently than when compared other methods whilst utilising large amounts of data available. Antonik *et al.* (2021), on the other hand, made use of Bayesian Optimisation for recurrent neural networks for Photonic Reservoir Computing (PRC). The paper concluded that the use of this method was extremely efficient and has the potential of becoming the standard for PRC hyperparameter selection in the future. In addition, Kandasamy *et al.* (2017) incorporated Bayesian Optimisation for hyperparameter selection for convolutional networks. The ability for Bayesian Optimisation to be used for sophisticated NN applications and its ability to increase efficiency and accuracy in hyperparameter selection when compared to alternate hyperparameter selection methods led to the increase in use of this method across different studies, industries and fields.

The three methods above were implemented for each model. Manual parameter selection, using trial and error and recommendations from research papers, was found to be beneficial in finding a decent performing basis model. This basis model was then used as a starting point and allowed for a good indication of the parameter ranges used in grid search. However, grid search was extremely time consuming. As the number of parameter combinations increased, the time taken to train the model exponentially increased. This method was therefore only feasible if a small number of parameter combinations were selected. This reduced the efficacy of grid search for this paper. In addition, grid search was completely uninformed of previous models trained and therefore spent a significant amount of time training models with sub-optimal hyperparameters. The Bayesian Optimisation training process was substantially faster than when compared to grid search. Therefore, the main focus of parameter tuning in this dissertation is using Bayesian Optimisation.

Bayesian Optimisation, like grid search, also made use of manual parameter selection to select well performing default hyperparameters into the Bayesian Optimisation algorithm. The Bayesian Optimisation algorithm was then conducted using the `Skopt` library (Louppe *et al.* (2020)) and outputted a model with parameter combinations that had a significantly lower loss metric than when compared to grid search for the Heston and SABR model under both methods. Therefore, Bayesian Optimisation for pricing and model calibration was explored in this dissertation and is explained using 5 steps.

Bayesian Optimisation Steps: (Bonald *et al.* (2020))
These steps are further explained using Figure 3.1 below:

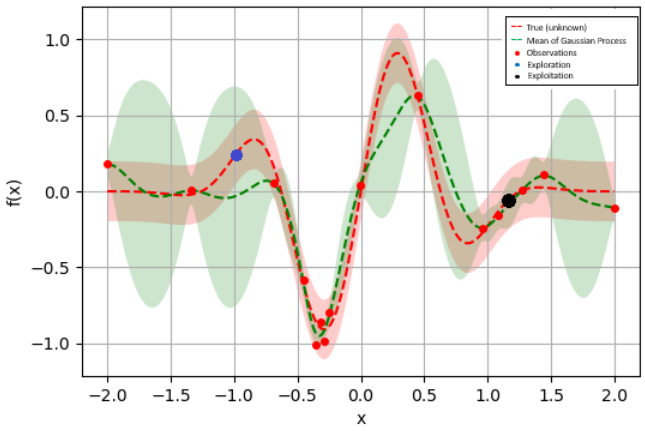

Fig. 3.1: Bayesian Optimisation using `Skopt` (Louppe *et al.* (2020)).

1. Begin with a few observations (illustrated by red dots on Figure 3.1) of the hyperparameters x_i and for the unknown objective function $f(x_i)$ for $i = 1, 2...n$.

2. Use the observations in step 1 to build a probabilistic model for the unknown

objective function f. The probabilistic model in this dissertation is a Gaussian process which constructs a joint probability distribution between the variables of interest. The Gaussian process regression model is then fit using a sample of inputs x and noisy evaluations for $f(x)$. This Gaussian process captures the behaviour of the sample and is set as the prior.

3. Since a Gaussian probabilistic function is used, one can compute the mean and standard deviation of the probabilistic function. The mean of the Gaussian prices is represented by the green dotted line in Figure 3.1 and the standard deviation is represented by the green shaded regions.

4. Select a location (for example either the blue or black dot in Figure 3.1) that is based on the posterior function. This location is selected by optimising an acquisition function U. This is mathematically expressed as:

$$x_{n+1} = \text{argmin}_x \, U(x)$$

The acquisition function, $U(x)$, which is computationally cheap to evaluate is used to select the points to evaluate. n represents the previous function evaluations that are used to update the prior and to form the posterior distribution. The acquisition function is designed to create (by optimising) a fine balance between exploration (improving areas of large uncertainty) and exploitation (reaching a location close to the value of the true function). In Figure 3.1, the black dot represents an area of exploitation where the mean of the Gaussian process is almost exactly equal to the true process. Conversely, the blue dot represents exploration where there is a large standard deviation represented by the shaded green region. The acquisition function $U(x)$ specifies what hyperparameter values x should be tried next with prior knowledge that x_n^* are the best hyperparameters observed thus far. Examples of acquisition functions include:

- Expected improvement:
  $-\mathbf{E}[f(x) - f(x_n^*)]$
- Lower confidence bound:
  $\mu_{GP}(x) + \kappa\sigma_{GP}(x)$, where κ controls the trade off between exploitation and exploration. $\mu_{GP}(x)$ represents areas for high exploitation and $\sigma_{GP}(x)$ represents areas of high exploration.
- Probability of improvement:
  $-P(f(x) \geq (x_m^*) + \kappa)$, where once again κ controls the trade off between exploitation and exploration.

5. Evaluate f and repeat steps 1 to 4. This process is repeated until all locations selected are close to the values of the true function.

This dissertation conducts the Bayesian Optimisation steps explained above using `gp_minimize` function in `python` available in the `Skopt` library. `gp_minimize` takes the objective function, upper and lower bounds for each model parameters, number of calls and acquisition function as inputs. The objective function is set

up to take in a list of hyperparameter values and returns the MSE loss of the ANN. Here the hyperparameters explored are learning rate, number of dense layers, number of nodes and whether dropout is present. The number of calls refers to the number of models you want to build to optimise the parameters and was set to 40. The acquisition function chosen was `gp_hedge`. Here, `gp_hedge` library was used to determine the best acquisition function to be selected at each iteration. The best acquisition function was chosen between the 3 functions discussed in step 4 above. The use of the `skopt` library provided an efficient method for hyperparameter selection whilst updating both the prior and posterior distribution to find optimal combinations of the hyperparameters of interest. Therefore, implementing the Bayesian Optimisation technique improves the accuracy and efficiency of selecting the best performing model.

3.7 Out-of-Sample Testing

The aim of this dissertation was to use an ANN trained (offline) to efficiently and accurately calibrate parameters and implied volatilities (online) for the SABR and Heston model. Therefore, it is important that this model performs well not only on the training and validation set but also on the unseen test set. Therefore, the model was tested on completely unseen data. Chapter 4 compares the results obtained using the unseen test set for each model and hence considers the use of these models for implied volatility and parameter calibration in a real world setting. The data, Bayesian Optimisation and best model ANN code can be found on Github: https://github.com/JennaGoosen/Volatility-Model-Pricing-Calibration-with-Neural-Networks-using-Bayesian-Optimisation

Chapter 4

Model Pricing & Calibration Results Under Indirect & Direct Method

This section analyses and compares the results obtained under the direct and indirect method for both the SABR and Heston model. The main focus of this chapter is the application of the direct and indirect method on calibrating the Heston model. Since the closed-form approximation of the SABR model can be computed extremely fast, even a well performing ANN is unable to outperform the speed of the closed form approximation of Hagan *et al.* (2002). Therefore, the SABR model is used as an initial experiment to verify the ability of ANN to accurately calibrate parameters.

For the indirect method, two sets of results are analysed. The first set of results explores the initial step of the indirect method which analyses the speed of the pre-trained ANN model to price the Heston and SABR model. The speed of the ANN model is compared to the computational time for the closed form approximation (for the SABR model) and characteristic pricing method followed by the Black-Scholes equation (for the Heston model) for implied volatility. The next set of results completes the indirect method, using a weighted least squares optimisation technique to calibrate model parameters by minimising the difference between the true and predicted volatilities, where the predicted implied volatilities are from the trained ANN (first step of indirect method). The weighted least squares optimisation procedure begins with creating a function with parameters as inputs and MSE as an output. This function follows the same data generation algorithm explained in Chapter 3 with the exception of step 1, where the parameters now are variables rather than randomly generated. The optimisation procedure was completed in `R Studio` using `optim`. "L-BFGS-B" method (Zhu *et al.* (1997)) which is a modification of the quasi-Newton method. This modification allows for a box constraint (where upper and lower bounds of the parameters are selected). The advantage of this method is its ability to solve non-linear problems. In addition, "L-BFGS-B" uses less memory than the BFGS quasi-Newton method. The next set of results analysed is the direct method. The direct methods accuracy and speed of using an ANN to output parameters given an implied volatility surface as inputs is analysed. The last section of results compares the indirect and direct approach

by using the predicted (calibrated) parameters to price the implied volatility function. Accuracy metrics are analysed to compare the performance of each method in pricing implied volatility.

For this dissertation, the weighted least squares optimisation algorithm was performed for completion of the pricing process. To accurately calibrate parameters using weighted least squares optimisation, one typically has to initialise the optimiser using a number of starting points for each parameter. This is done to increase the probability of the optimal parameters reaching a global minima. Complex loss functions (like in the case of the SABR and Heston model) often have multiple local minimas. Therefore, certain modifications of the loss function are required (for example, the use of a penalty function to increase convexity) to ensure the objective function is more easily minimised. Consequently, for the indirect pricing model to accurately calibrate model parameters one is typically required to run the algorithm multiple times. This increases calibration time. Due to time and content limitations, this dissertation only performed the weighted least squares optimisation algorithm once, with initial parameter values set equal to their means. As a result, calibration for the indirect method was conducted for completion purposes and was not perfected in this dissertation. The results are displayed in Appendix B. Therefore, the focus of results explored for this dissertation are set out as follows:

- The use of ANNs to speed up the initial step of the indirect method.
  - The comparison of efficiency of pricing using ANNs versus traditional pricing methods such as characteristic pricing functions and closed form approximations.
- The ability of ANNs to calibrate model parameters accurately using the direct method.
- The comparison of the indirect (using a single run of the weighted least squares optimisation algorithm) and direct method for calibration and the use of these calibrated parameters to then price implied volatilities.

The methodology explained in Chapter 3 was implemented and the results for each model and method are analysed and discussed below.

4.1 SABR Model Calibration and Pricing

As mentioned above, the SABR model is explored to verify the ability of an ANN to price and calibrate parameters.

4.1.1 Indirect Pricing and Calibration Results

The indirect pricing method uses parameters as inputs to price an implied volatility surface (outputs) over a range of strike prices. After which, a traditional least squares optimisation approach was implemented to calibrate the SABR model parameters ν, α and ρ. These parameters are then plugged back into the implied volatility pricing function to investigate their pricing accuracy.

Indirect SABR Pricing Results

The first step of the indirect method uses an ANN to price options. Optimal parameters were selected after running Bayesian Optimisation. The following parameters were found to produce the model with the lowest mean square error:

Learning Rate	No. of Layers	No. of Nodes	Batch Size	Dropout
1e-02	5	204	16	False

Tab. 4.1: Optimal Hyperparameters for the SABR Pricing Model.

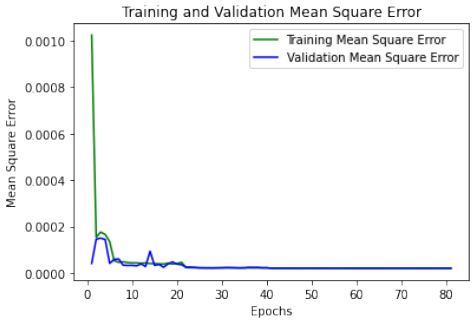

Fig. 4.1: SABR Implied Volatility MSE.

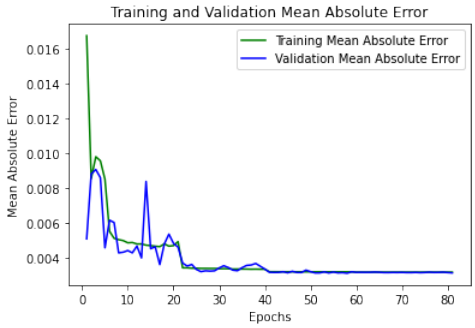

Fig. 4.2: SABR Implied Volatility MAE.

Figure 4.1 and Figure 4.2 plot the MSE and MAE across 84 epochs. After 84 epochs the early stopping criteria stopped training to ensure that the training dataset did not overfit. Figure 4.1 and Figure 4.2 exhibit a staggered decreasing trend for the first 20 epochs followed by a gradual decrease for the remaining epochs. The MSE and MAE trends explained above are relatively consistent for both the training and validation loss and level out to the same values when early stopping occurs.

Figures A.1 to A.6 in Appendix A illustrates the performance of the SABR pricing model. These figures compare the true simulated implied volatilities against the predicted implied volatilities on the unseen test set. The perfect linear relationship between the true and predicted implied volatilities for all 36 figures indicates that the deep NN was able to accurately capture the complex relationships between the input and outputs. This is also proven by observing MSE, MAE and the relative error $\left(\frac{\text{true}-\text{predicted}}{\text{true}}\right)$ of 1000 unseen samples. The pre-trained ANN resulted in an MSE of 2.0384e-05, MAE of 0.0032 and relative error of 7.30e-04 on 1000 unseen samples.

However, as mentioned above, the closed form approximation of the SABR model is extremely efficient. An Intel Core i5-6300 CPU @ 2.40GHz with 2 cores and 4 logical processors and 8GB of RAM was used to compare the speed of the closed form approximation and the ANN for a sample of 1000. The time of the ANN to predict the implied volatility surface for a sample of 1000 was 130.3261 seconds, whilst the closed form approximation generated the implied volatility surface in

0.1153 seconds. Therefore, one is better off using the closed form approximation for SABR model pricing than compared to a 5 hidden layer deep NN (found using Bayesian Optimisation).

Indirect SABR Calibration Results

Even though the closed form approximation was substantially faster than the pre-trained deep NN, the indirect method of SABR model calibration was still investigated. The predicted implied volatility surfaces outputted (from indirect pricing) and the true implied volatility surfaces were used to calibrate the model. A single run through of the weighted least squares algorithm was implemented. This was done using `R Studio's optim` function. The lower and upper bounds of the parameters ν, α and ρ (Table 3.1) were plugged into `optim`. Initial parameter values for ν, α and ρ were set equal to their true means. Parameter α was predicted relatively well. However, this algorithm failed to produce accurate values for ν and ρ. These results are found in Figure B.1 in Appendix B. The failure of this calibration of parameters ν and ρ was due to the poor selection of initial parameter values. Therefore, in order to accurately calibrate parameters one would require the weighted least squares optimisation to be augmented by a penalty function with multiple initial parameter values.

For completion of the pricing process, the "incorrect" model parameters found (using a single run through of the weighted least squares algorithm) were then plugged back into the pricing function and produced prices that were very similar to true prices. The MSE for 1000 predicted prices versus true prices was 1.26e-05. In addition, the true implied volatilities for 1 year ATM option were compared to the predicted implied volatilities. Figure 4.3 below exhibits a clear linear relationship between the true and predicted implied volatilities for 1 year ATM options. However, there are slightly greater errors for higher implied volatilities than lower implied volatilities.

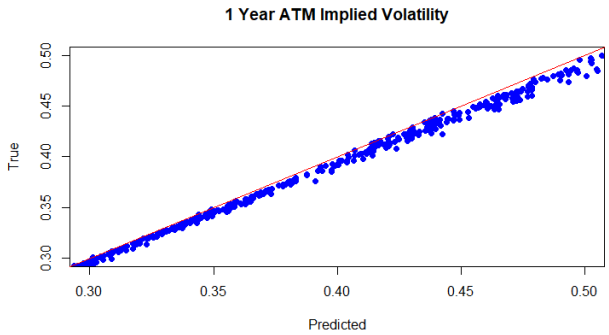

Fig. 4.3: Indirect Method: True vs Predicted SABR 1 Year ATM Option.

Overall, the use of ANNs was not advantageous for the SABR model for the pricing step of the indirect method. This was due to the ability of the closed form approximation to rapidly produce the implied volatility surface. However, the ANN was still extremely accurate for pricing using the indirect method. In addition, the calibrated parameters from the second step of the indirect method were able to price ATM options well.

Direct SABR Calibration Results

The direct model calibration for the SABR model involved using an implied volatility surface for a range of strikes as inputs to directly calibrate the SABR model parameters as outputs. Optimal parameters were found by training each parameter separately.

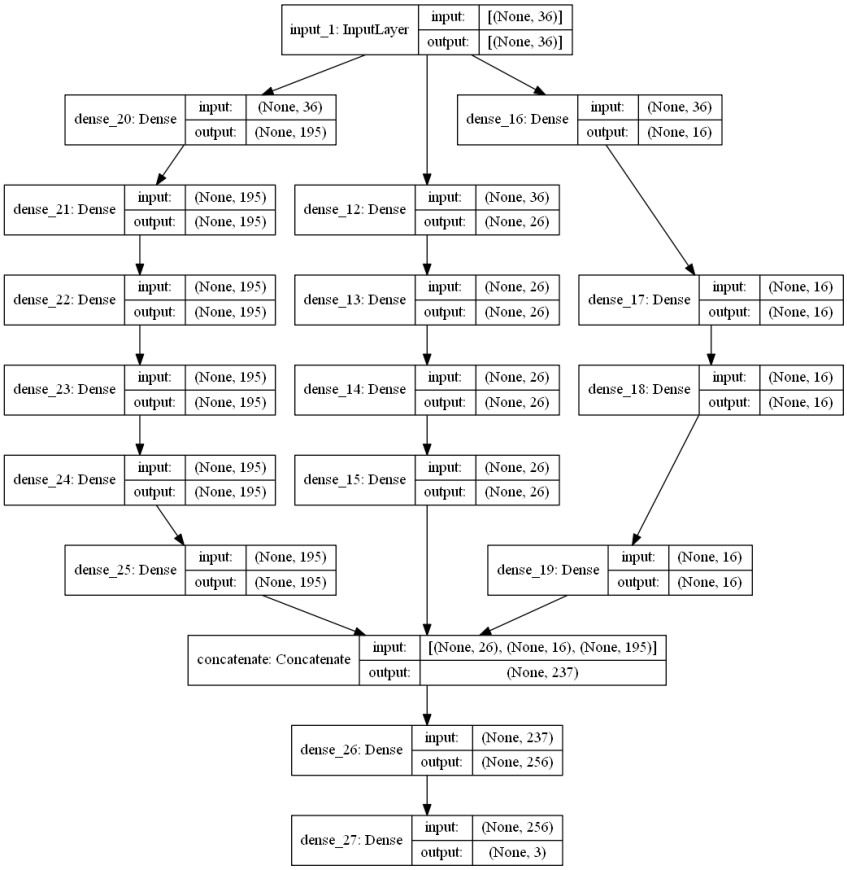

Fig. 4.4: SABR Neural Network Model Graph.

Figure 4.4 illustrates the process of learning the SABR model parameters ν, α and ρ separately. The deep NN began with an input layer consisting of all 36 implied volatility inputs. After which, the ANN was split into three branches for each parameter to train separately.

Bayesian Optimisation was used to determine the number of layers required to train each parameter and the number of nodes within each layer. Similarly to the indirect method, the learning rate, batch size and dropout are also selected using Bayesian Optimisation. The optimal hyperparameters selected were:

Learning Rate	No. of Layers			No. of Nodes			Batch Size	Dropout
2e-05	ν: 4	α: 4	ρ: 6:	ν: 26	α: 16	ρ: 195	216	False

Tab. 4.2: SABR Direct Method Optimal Hyperparameters.

The model was trained using 362 epochs. After 362 epochs the early stopping criteria was reached and the model training was stopped to prevent overfitting. Figure 4.5 and 4.6 differ from the figures obtained under indirect method. These figures display a steep decrease for the first 10 epochs followed by a smooth decrease for the remaining epochs. Once again the training and validation loss performed similarly.

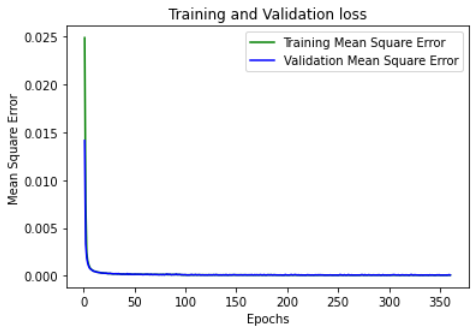

Fig. 4.5: SABR Parameter MSE.

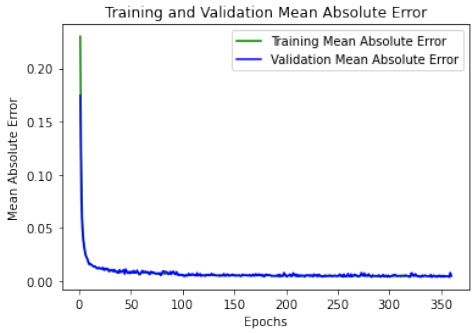

Fig. 4.6: SABR Parameter MAE.

Similarly to the indirect method, an unseen sample of 1000 is plugged into the pre-trained model to test the performance of the model in predicting parameters ν, α and ρ. It is clear from Figure 4.7 that both ν and α are accurately trained. However, ρ appears to be less accurate. The customised weighted loss function applied heavier weighting to the parameter ρ which improved the accuracy of training substantially than the standard MSE loss function. The overall MSE was 2.47e-04 and the MAE was 0.0084.

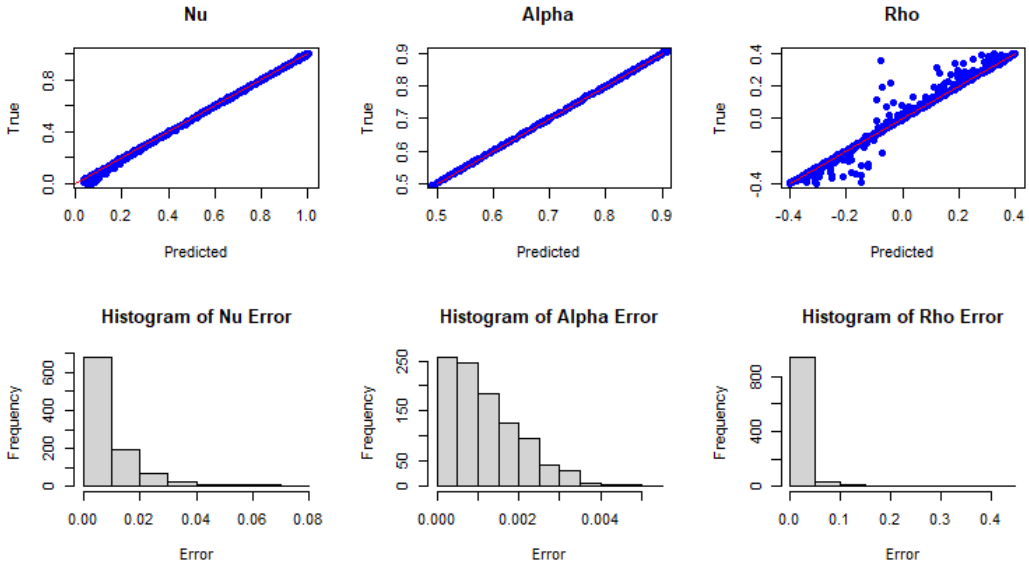

Fig. 4.7: Direct Method: True vs Predicted SABR Parameters.

This is further seen by Table 4.3 which displays the model calibration results for the first three unseen samples.

	Output 1		Output 2		Output 3	
Par.	True	Direct	True	Direct	True	Direct
ν	0.6752	0.6764	0.6576	0.6562	0.9143	0.9106
α	0.3905	0.3886	0.3883	0.3896	0.7333	0.7331
ρ	0.2959	0.2909	0.1763	0.1743	0.2684	0.2694

Tab. 4.3: SABR Indirect vs Direct Method.

From the linear plots, error histograms and the table of parameter values outputted using the direct method, it is clear that the ANN was able to accurately predict model parameters using an implied volatility surface over a range of strikes as inputs. The performance of the model online was then further explored. The predicted parameters were plugged into the pricing function and implied volatility surfaces were computed for all 1000 parameter samples. The true implied volatility was then compared to the predicted implied volatility using the calibrated parameters. The MSE obtained was 1.896171e-04. In addition, the 1 year ATM option price was computed using the pre-trained model. Figure 4.8 illustrates the true versus predicted implied volatility for 1 year options ATM. It is clear that there is a linear relationship between the true and predicted implied volatilities. Similarly to the indirect method, the error is smaller for lower implied volatility values and slightly larger the higher the implied volatility.

Overall, the true versus predicted parameters are relatively similar. This suggests that the model was able to capture the relationship between the volatility sur-

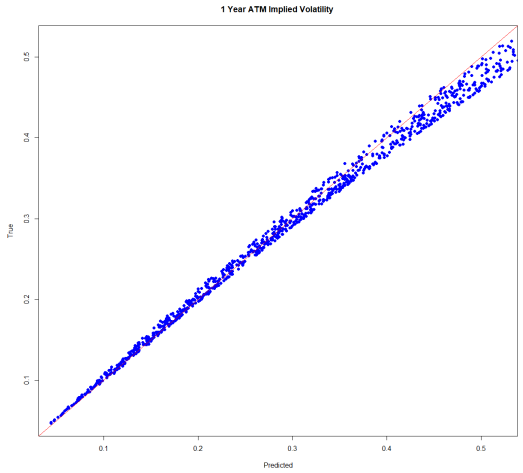

Fig. 4.8: Direct Method: True vs Predicted SABR 1 Year ATM Option.

face and parameters. However, predictions for parameter ρ deviate slightly from the true values. Therefore, this model should be used with caution in a real world context. Even though parameter ρ was not perfectly calibrated, when the parameters were plugged back into the pricing function, accurate implied volatilities were produced.

Indirect Versus Direct SABR Model Calibration Comparison

The indirect method was substantially more computationally expensive on unseen data. The entire direct model calibration procedure was completed in 123.8229 seconds for a sample of 1000. This is faster than the first step of the ANN to predict the pricing function which takes 130.3261 seconds. In addition, to accurately calibrate parameters for the indirect method, one would typically have to run the optimiser multiple times which makes the direct method even more attractive. However as mentioned previously, the use of ANNs (for both indirect and direct methods) was still substantially slower than using the closed form approximation. This calibration process was used as an experiment to see whether ANNs were able to accurately calibrate models. The next section in the results will explore ANNs for the indirect and direct method on the more complex Heston model.

4.2 Heston Model Calibration and Pricing

As previously mentioned, there are no readily available closed form approximations for the Heston model. Therefore, option prices are calculated using the Little Heston Trap formulation of the Heston characteristic function to price options. These option prices are then set equal to Black-Scholes option prices to solve for the implied volatility. This procedure made the pricing and calibration of the Heston model extremely time consuming. Therefore, the use of ANNs under the indirect and direct method were explored and analysed.

4.2.1 Heston Indirect Pricing and Calibration Results

The first step of the indirect method used an ANN with parameters as inputs and predicted a volatility surface over a range of strikes. The second step then used a weighted least squares optimisation approach to calibrate the model and predict values of $\nu_0, \kappa, \theta, \sigma$ and ρ. After which, these parameters were then used in the pricing function to compare the accuracy of the predicted option price against the true option price.

Indirect Heston Pricing Results

Similarly to the SABR pricing ANN, the optimal Heston model was found using Bayesian Optimisation. The optimal hyperparameters that produced the lowest error were:

Learning Rate	No. of Layers	No. of Nodes	Batch Size	Dropout
2e-05	3	270	45	False

Tab. 4.4: Optimal Hyperparameters for the Heston Pricing Model.

The training and validation MSE and MAE are tracked for each epoch (Figure 4.9 and Figure 4.10). Once again a very steep decrease of both metrics was observed for the first 2 epochs (Figure 4.9 and Figure 4.10), with a gradual decrease for the remaining epochs. After 479 epochs the early stopping criteria was reached.

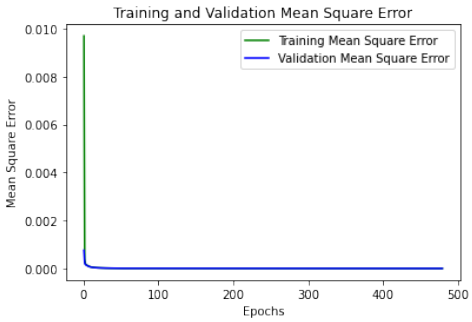

Fig. 4.9: Heston Implied Volatility MSE.

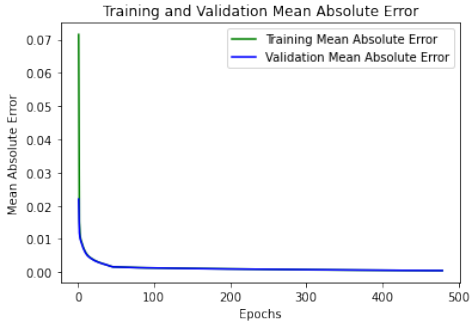

Fig. 4.10: Heston Implied Volatility MAE.

Figure A.7 to A.12 in Appendix A, illustrates the models ability to capture the complex relationship between raw outputs and inputs with 1000 unseen samples. The figures illustrate an almost perfect linear relationship between the true and predicted Heston implied volatilities. The Heston volatility model on 1000 unseen samples had a MSE of 1.0890e-06, MAE of 7.1103e-04 and a relative error of 2.63e-06.

The ANN was substantially faster to predict 1000 samples than compared to the simulated implied volatilities using the Little Heston Trap formulation of the characteristic function. The ANN took 127.0258 seconds while the simulated implied

volatilities took 454.9504 seconds. Therefore, unlike the SABR model, it is plausible to make use of ANN for model calibration.

Indirect Heston Calibration Results

The implied volatility surface generated using the pre-trained ANN was used to calibrate the model for 1000 unseen samples using `R Studio's optim` function. The lower and upper bounds of the parameters $\nu_0, \kappa, \theta, \sigma$ and ρ (Table 3.2) were plugged into `optim`. Initial parameter values were set equal to their true means. Once again, only one run of the weighted least squares algorithm was implemented. This method was able to calibrated parameter θ accurately, however, was unable to predict ν_0, κ, σ and ρ accurately. These results are found in Figures B.2 and B.3 in Appendix B. Similarly to the SABR indirect method second step, the failure of this calibration of parameters was due to the poor selection of initial parameter values and omission of the penalty function. Therefore, to accurately calibrate the Heston model parameters using this method, one would require more sophisticated methods of initial parameter selection which will in turn increase the computation time of the algorithm.

In a similar manner to the SABR model, the poor calibrated model parameters when plugged back into the pricing function produced prices that are very similar to the true prices. The MSE for 1000 predicted prices versus true prices was 0.0012 and the relative error was 0.0173. In addition, the true implied volatility was compared to the predicted implied volatility for 1 year ATM options. Figure 4.11 below exhibits a clear linear relationship between the true and predicted implied volatilities for 1 year ATM options.

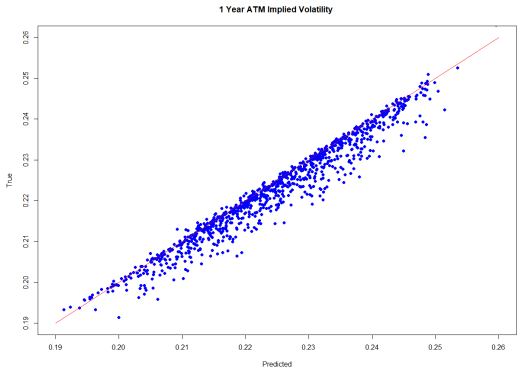

Fig. 4.11: Indirect Method: True vs Predicted Heston 1 Year ATM Option.

In conclusion, the use of ANNs for pricing under the indirect method is advantageous. This is due to the ANN learning the pricing function more rapidly than when compared to the Little Heston Trap characteristic pricing method. Model calibration of the Heston model under the indirect method failed to calibrate the model parameters accurately using a single run of the weighted least squares algorithm. However, the parameters found using the indirect method were still able to price options relatively accurately. Therefore, to improve calibration using the indirect method, one should use an ANN for the first step followed by a weighted least

squares algorithm with multiple initial parameter values and penalty function.

Direct Heston Calibration Results
The optimal model hyperparameters found using Bayesian Optimisation for the Heston model using the direct method are:

Learning Rate	No. of Layers	No. of Nodes	Batch Size	Dropout
7e-03	5	250	45	False

Tab. 4.5: Heston Direct Method Optimal Hyperparameters.

The direct method for the Heston Model differed from the SABR model since the parameters were not trained separately for the Heston model. The training and validation loss for the first 15 epochs was unstable and then began to decrease substantially and smoothen for the remaining epochs. For this model, the early stopping criterion was activated after 136 epochs.

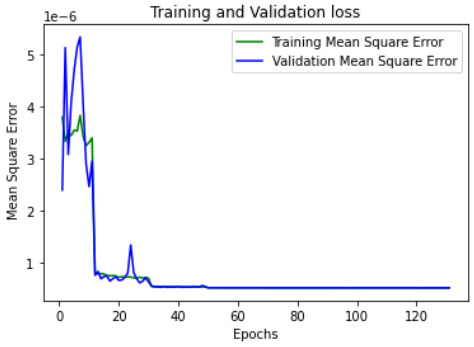

Fig. 4.12: Heston Parameter MSE.

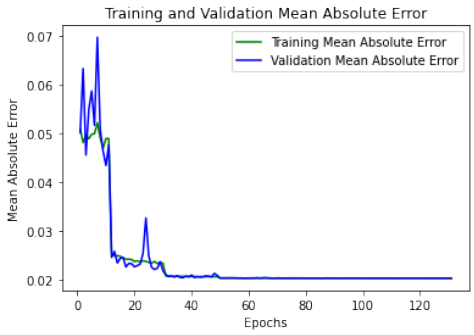

Fig. 4.13: Heston Parameter MAE.

The overall MSE for the 1000 unseen samples was 5.3506e-07 and the MAE was 0.0205. This was substantially improved when the customised weighted loss function was used. Figure 4.14 and 4.15 illustrates the linear relationship between the true and predicted parameter outputs. It is clear that all five parameters exhibit a linear relationship between the true and predicted. This is further illustrated by the histogram error plots with high frequencies for very small errors.

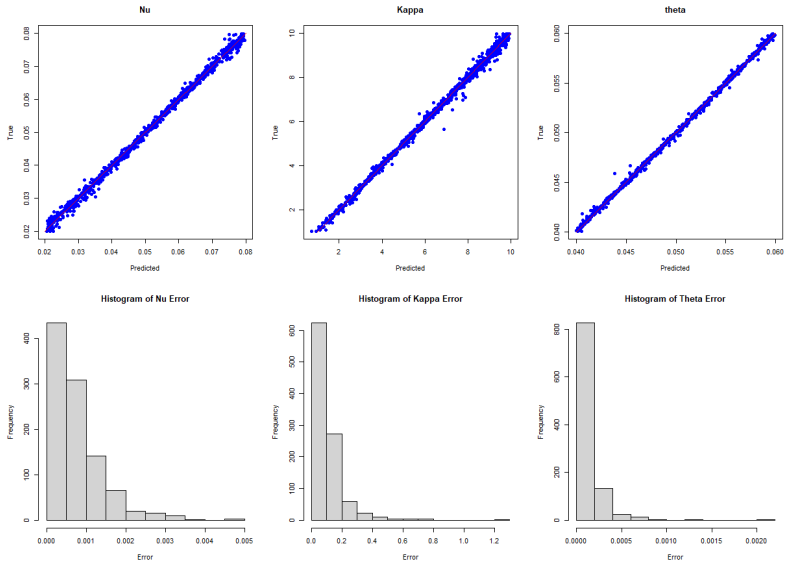

Fig. 4.14: Direct Method: True vs Predicted Heston Parameters ν_0, κ and θ.

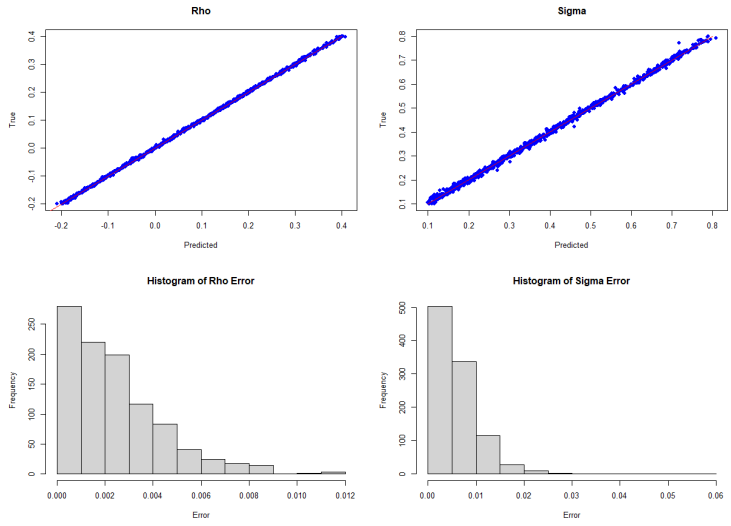

Fig. 4.15: Direct Method: True vs Predicted Heston Parameters σ and ρ.

This is further seen by Table 4.6 which displays the model calibration results for the first three unseen samples.

	Output 1		Output 2		Output 3	
Par.	True	Direct	True	Direct	True	Direct
ν_0	0.0301	0.0313	0.0512	0.0524	0.0374	0.0375
κ	9.9768	8.2530	8.0694	7.8714	9.3231	9.4360
θ	0.0407	0.0395	0.0593	0.0589	0.0531	0.0530
σ	0.1396	0.1289	0.6603	0.6196	0.7165	0.6855
ρ	-0.1511	-0.1165	0.1892	0.1728	0.0196	0.0099

Tab. 4.6: Heston Indirect vs Direct Method.

From the linear plots, error histograms and the table of parameter values outputted using the direct method, it is clear that the ANN is able to accurately predict model parameters using an implied volatility surface over a range of strikes as inputs. The predicted parameters where then plugged back into the option pricing function and the implied volatilities obtained were compared to the true implied volatilities. The total MSE and mean relative error for the 1000 unseen samples was 7.078483e-05 and 0.01553519. To further test the performance of pre-trained model parameter calibration, the calibrated parameters were used to predict implied volatilities for a 1 year ATM options. Figure 4.16 illustrates the linear relationship between the true and predicted ATM options.

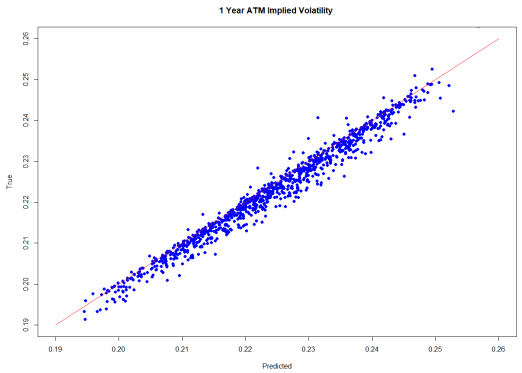

Fig. 4.16: Direct Method: True vs Predicted Heston 1 Year ATM Option.

Therefore, the deep NN trained using the implied volatility surface over a range of strikes as inputs was able to accurately learn the model parameters $\nu_0, \kappa, \theta, \rho$ and σ. In connection, the parameters obtained were then used to accurately price options.

Indirect Versus Direct Heston Model Calibration Comparison

Conversely to the SABR parameter calibration results, the use of ANNs for both methods is advantageous. This is due pricing using the indirect method being substantially faster than using the Little Heston Trap followed by solving the Black-Scholes equation to obtain implied volatilities. The results also found the direct

method to outperformed the indirect method. This was seen not only by the ability of the direct method to accurately predict the model parameters, but it was also proven by the substantially lower MSE when using the predicted parameters to price the option. In addition, the first step of the indirect method (the ANN to predict the implied volatility surface) had a computational time of 127.1969 seconds whilst the entire direct method had a computational time of 127.0258 seconds on the 1000 unseen samples. In addition, more computational time would be required to accurately calibrate the parameters under the indirect method using multiple initial parameter values and incorporating a penalty function. Therefore, the direct method was computationally more efficient even before computing the second step of the indirect method. Overall, deep NN were able to accurately price and calibrate the Heston model.

Chapter 5

Conclusion

This dissertation found that ANNs have the ability to efficiently and accurately price and calibrate complex stochastic volatility models. The use of Bayesian Optimisation for hyperparameter selection was able to substantially speed up the process of selecting the best model than when compared to grid search and manual selection. In addition, Bayesian Optimisation models selected exhibited a substantially lower loss metric than when compared to manual selection and grid search. However, less complex volatility models (such as the SABR model) where closed form solutions are available are able to outperform the speed of a well performing ANN.

Even though the ANN was outperformed by the closed form solution for the SABR model, pricing and calibration under the indirect and direct method was still successful. The direct method was able to calibrate parameters of ν and α with high accuracy. Calibration of parameter ρ was not as accurate. Possible improvements of this calibration procedure would be to redefine the bounds for ρ. Because of the leverage effect, one generally expects ρ to be less than zero. Therefore, the data generation and model calibration procedure could be to constrain ρ such that it is less than zero.

When comparing the indirect and direct method for the Heston model, the direct method was substantially faster. The indirect method calibration process using a single run through of the weighted least squares algorithm was sensitive to the initial parameters selected. Therefore, to improve these results further research involves adapting the optimiser by incorporating a penalty function and running the optimiser multiple times. However, the aim of this dissertation explored whether ANNs were able to accurately and efficiently price and calibrate using the indirect and direct method. The results proved that the use of ANNs for pricing using the indirect method and calibration for the direct method, was very accurate and efficient when tested on unseen samples for the Heston model. The indirect method was compared to a traditional calibration method. This calibration method involved using Black-Scholes equation and the Heston characteristic pricing method to solve for 1000 implied volatilities and then using these for least squares optimisation for model calibration. The trained ANN was able to generated 1000 predicted implied volatilities 3.58 times faster than when compared to solving for implied volatilities using Black-Scholes equation and the Heston characteristic pricing method. Therefore, unlike the SABR model, it is plausible to make use of ANN for model calibration. Under both the indirect and direct method, the Hes-

ton ANN model selected using Bayesian Optimisation produced very good results when compared to the traditional method.

Future research includes implementing various methods of pricing and comparing the speed and accuracy directly to the pricing step of the indirect method. In addition, the Bayesian Optimisation algorithm can be run to tune for more hyperparameters. For example, this dissertation used a weighted loss function which substantially decreased the loss of the best performing models. Therefore, Bayesian Optimisation can be used to tune the weights associated with each parameters contribution to the overall loss function.

The generalisation of the calibrated parameter to then price the implied volatility surface was also accurate for both the direct and indirect methods. The dissertation further explored the ability of the pre-trained ANN to predict 1 year ATM options. In reality, performance of ATM options is more important than options deeply in or out the money. Both the models under the direct and indirect methods were tested for their ability to price a 1 year ATM option. The results indicated that these models were able to accurately price these options. The direct method was able to not only accurately price 1 year ATM options but was able to do this extremely fast. Therefore, the use of ANNs where the model is pre-trained offline and used to price online (especially using the direct method) is extremely efficient and accurate and should be used for complex models where closed form approximations are not readily available.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J. and Devin, M. *et al*. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems, *arXiv preprint arXiv:1603.04467* .

Abu-Mostafa, Y. S. (2001). Financial model calibration using consistency hints, *IEEE transactions on neural networks* **12**(4): 791–808.

Albrecher, H., Mayer, P., Schoutens, W. and Tistaert, J. (2007). The Little Heston Trap, *Wilmott* (1): 83–92.

Antonik, P., Marsal, N., Brunner, D. and Rontani, D. (2021). Bayesian Optimisation of large-scale photonic reservoir computers, *Cognitive Computation* pp. 1–9.

Bennell, J. and Sutcliffe, C. (2004). Black–Scholes versus artificial neural networks in pricing FTSE 100 options, *Intelligent Systems in Accounting, Finance & Management: International Journal* **12**(4): 243–260.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization., *Journal of machine learning research* **13**(2).

Black, F. (1976). Studies of stock market volatility changes, *1976 Proceedings of the American statistical association business and economic statistics section* .

Bonald, T., de Lara, N., Lutz, Q. and Charpentier, B. (2020). Scikit-network: Graph analysis in python., *J. Mach. Learn. Res.* **21**: 185–1.

Brigo, D., Huang, X., Pallavicini, A. and Sáez de Ocáriz Borde, H. (2021). Interpretability in deep learning for finance: a case study for the Heston model, *Available at SSRN 3829947* .

Chan, J. H. and Joshi, M. S. (2010). *Fast and accurate long stepping simulation of the Heston stochastic volatility model*, Centre for Actuarial Studies, Fac. of Economics & Commerce, University of Melbourne.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function, *Mathematics of control, signals and systems* **2**(4): 303–314.

Dupire, B. *et al*. (1994). Pricing with a smile, *Risk* **7**(1): 18–20.

Gil-Pelaez, J. (1951). Note on the inversion theorem, *Biometrika* **38**(3-4): 481–482.

Gupta, A. (2010). *A Bayesian approach to financial model calibration, uncertainty measures and optimal hedging*, PhD thesis, Oxford University, UK.

Hagan, P. S., Kumar, D., Lesniewski, A. S. and Woodward, D. E. (2002). Managing smile risk, *The Best of Wilmott* **1**: 249–296.

Hasanhodzic, J. and Lo, A. W. (2011). Black's leverage effect is not due to leverage. Available at SSRN: https://papers.ssrn.com/sol3/papers.cfm?abstract$_id = 1762363[04/01/2022]$.

Hernandez, A. (2016). Model calibration with neural networks, *Risk* .

Heston, S. (1993). 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.

Horvath, B., Muguruza, A. and Tomas, M. (2021). Deep learning volatility: a deep neural network perspective on pricing and calibration in (rough) volatility models, *Quantitative Finance* **21**(1): 11–27.

Hu, J. (2013). *Barrier Option Pricing under SABR model Using Monte Carlo methods*, PhD thesis, Worcester Polytechnic Institute. Available at: https://digital.wpi.edu/concern/etds/nc580m77 [03/01/2022].

Huang, J., Chai, J. and Cho, S. (2020). Deep learning in finance and banking: A literature review and classification, *Frontiers of Business Research in China* **14**: 1–24.

Huang, X. (2020). Interpretability in deep learning for Heston smile modeling calibration. Available at: https://www.imperial.ac.uk/media/imperial-college/faculty-of-natural-sciences/department-of-mathematics/math-finance/Xiaoshan_Huang_Thesis.pdf [04/01/2022].

Joy, T. T., Rana, S., Gupta, S. and Venkatesh, S. (2016). Hyperparameter tuning for big data using Bayesian Optimisation, *2016 23rd International Conference on Pattern Recognition (ICPR)*, IEEE, pp. 2574–2579.

Kandasamy, K., Krishnamurthy, A., Schneider, J. and Poczos, B. (2017). Asynchronous parallel Bayesian Optimisation via thompson sampling, *arXiv preprint arXiv:1705.09236* .

Kim, H. Y. and Won, C. H. (2018). Forecasting the volatility of stock price index: A hybrid model integrating lstm with multiple garch-type models, *Expert Systems with Applications* **103**: 25–37.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980* .

Liu, S., Borovykh, A., Grzelak, L. A. and Oosterlee, C. W. (2019). A neural network-based framework for financial model calibration, *Journal of Mathematics in Industry* **9**(1): 1–28.

Louppe, G., Kumar, M. and Nahrstaedt, H. (2020). Bayesian Optimization with skopt, *Scikit-Optimize contributors (BSD License).* Available at: https://scikit-optimize.github.io/0.8/auto$_{e}xamples/bayesian - optimization.html$[12/01/2022].

McGhee, W. (2020). An artificial neural network representation of the SABR stochastic volatility model, *Journal of Computational Finance* **25**(2).

Mostafa, B., El-Attar, N., Abd-Elhafeez, S. and Awad, W. (2020). Machine and deep learning approaches in genome: Review article, *Alfarama Journal of Basic Applied Sciences* .

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N. and Antiga, Luca *et al*, j. v. p. (2019). Pytorch: An imperative style, high-performance deep learning library.

Roeder, D. and Dimitroff, G. (2020). Volatility model calibration with neural networks a comparison between direct and indirect methods, *arXiv preprint arXiv:2007.03494* .

Rönnqvist, S. and Sarlin, P. (2017). Bank distress in the news: Describing events through deep learning, *Neurocomputing* **264**: 57–70.

Sehgal, N. and Pandey, K. K. (2015). Artificial intelligence methods for oil price forecasting: a review and evaluation, *Energy Systems* **6**(4): 479–506.

Sevim, C., Oztekin, A., Bali, O., Gumus, S. and Guresen, E. (2014). Developing an early warning system to predict currency crises, *European Journal of Operational Research* **237**(3): 1095–1104.

Shen, F., Chao, J. and Zhao, J. (2015). Forecasting exchange rate using deep belief networks and conjugate gradient method, *Neurocomputing* **167**: 243–253.

Song, Q., Liu, A. and Yang, S. Y. (2017). Stock portfolio selection using learning-to-rank algorithms with news sentiment, *Neurocomputing* **264**: 20–28.

Thorin, H. (2020). Artificial neural networks for SABR model calibration & hedging. Available at: https://www.imperial.ac.uk/media/imperial-college/faculty-of-natural-sciences/department-of-mathematics/math-finance/Thorin_MSc_Thesis_NN_SABR.pdf [04/01/2022].

Tripathy, S. and Rahman, A. (2013). Forecasting daily stock volatility using GARCH model: A comparison between BSE and SSE, *IUP Journal of Applied Finance* **19**(4): 71.

Zhu, C., Byrd, R. H., Lu, P. and Nocedal, J. (1997). Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization, *ACM Transactions on mathematical software (TOMS)* **23**(4): 550–560.

Zhylyevskyy, O. (2012). Efficient pricing of european-style options under Heston's stochastic volatility model.

Appendix A

Indirect Method: True Versus Predicted Implied Volatilities

A.1 Indirect Method Implied Volatilities for SABR and Heston Model

The figures below display the true versus predicted implied volatilites under the indirect method for the SABR and Heston Model.

A.1.1 SABR Implied Volatilities True Versus Predicted

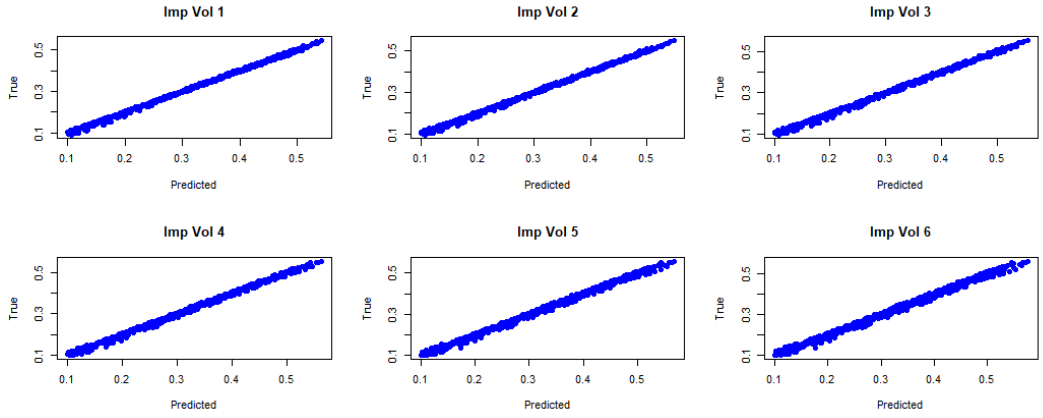

Fig. A.1: True vs Predicted (SABR) Implied Volatilities (1 to 6).

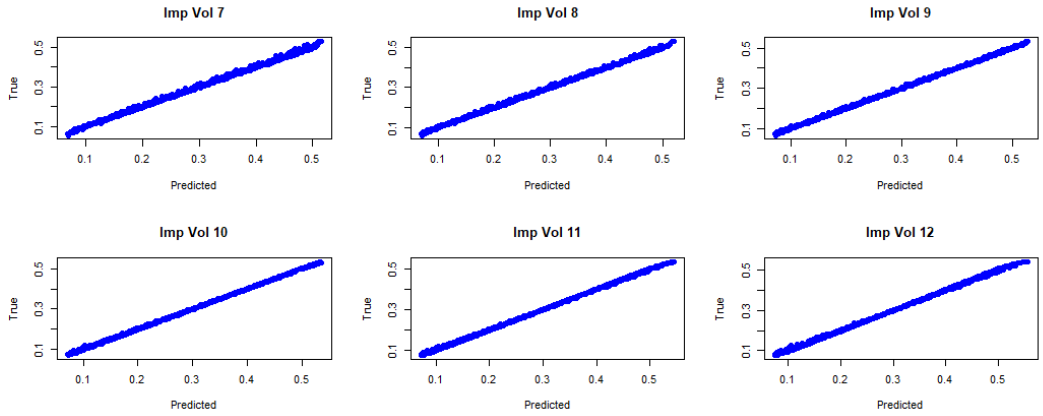

Fig. A.2: True vs Predicted (SABR) Implied Volatilities (7 to 12).

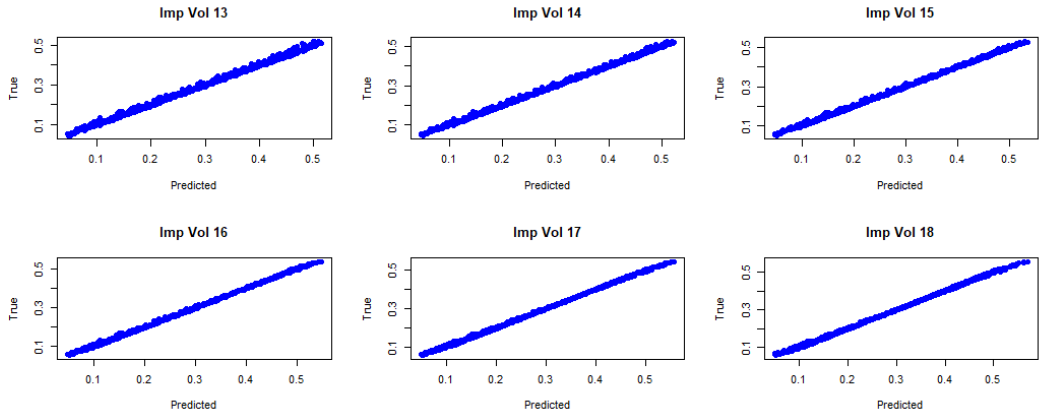

Fig. A.3: True vs Predicted (SABR) Implied Volatilities (13 to 18).

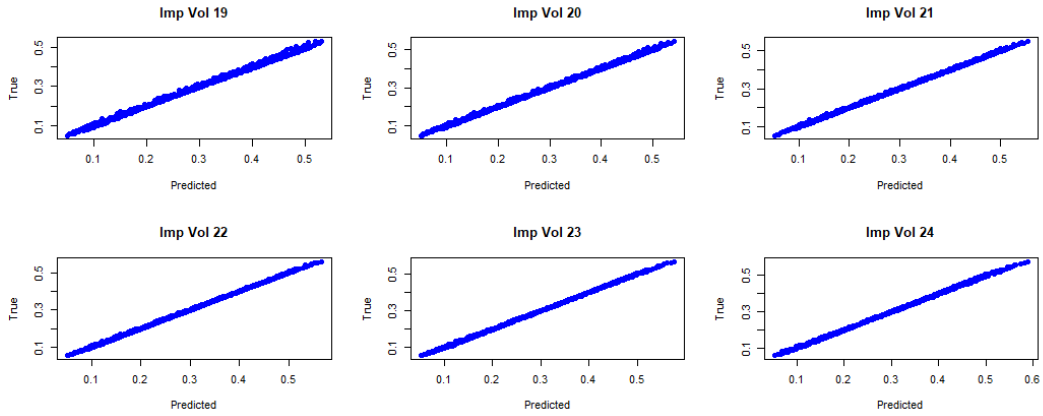

Fig. A.4: True vs Predicted (SABR) Implied Volatilities (19 to 24).

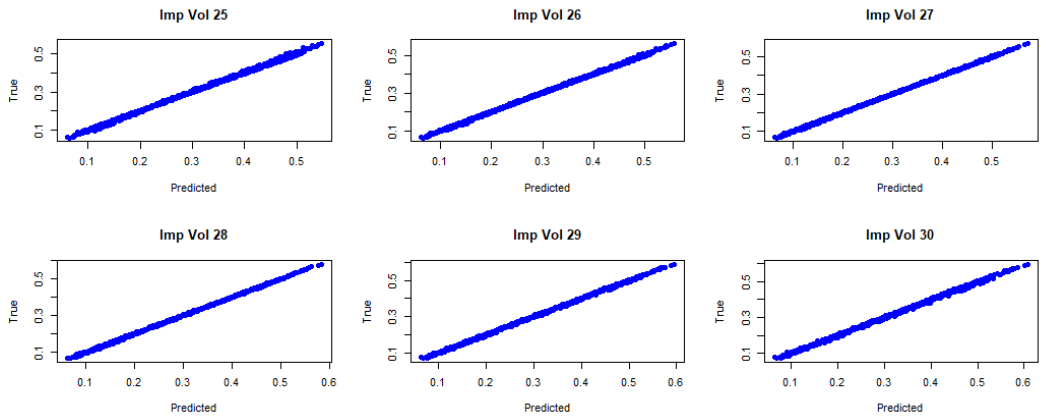

Fig. A.5: True vs Predicted (SABR) Implied Volatilities (25 to 30).

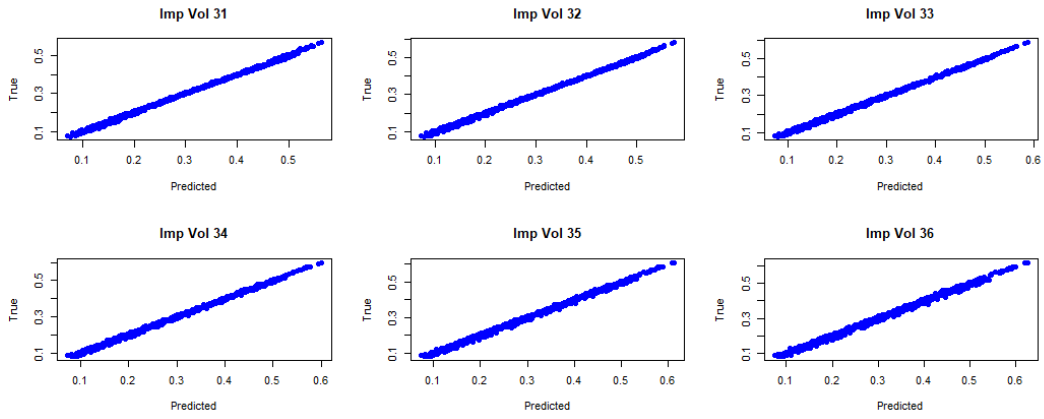

Fig. A.6: True vs Predicted (SABR) Implied Volatilities (31 to 36).

A.1.2 Heston Implied Volatilities True Versus Predicted

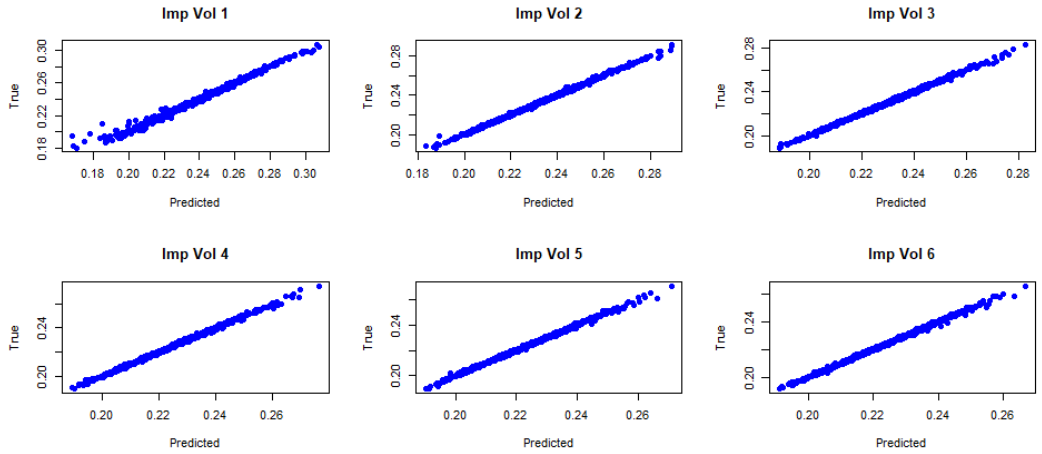

Fig. A.7: True vs Predicted (Heston) Implied Volatilities (1 to 6).

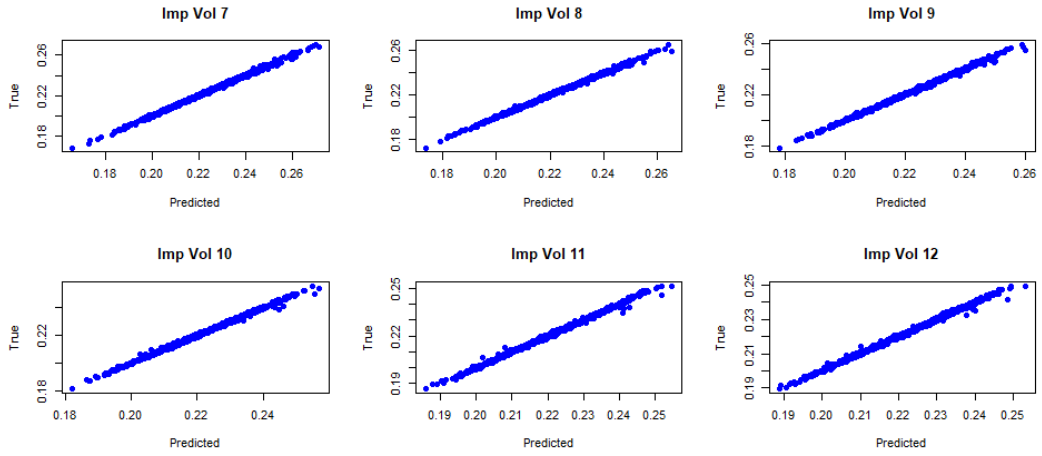

Fig. A.8: True vs Predicted (Heston) Implied Volatilities (7 to 12).

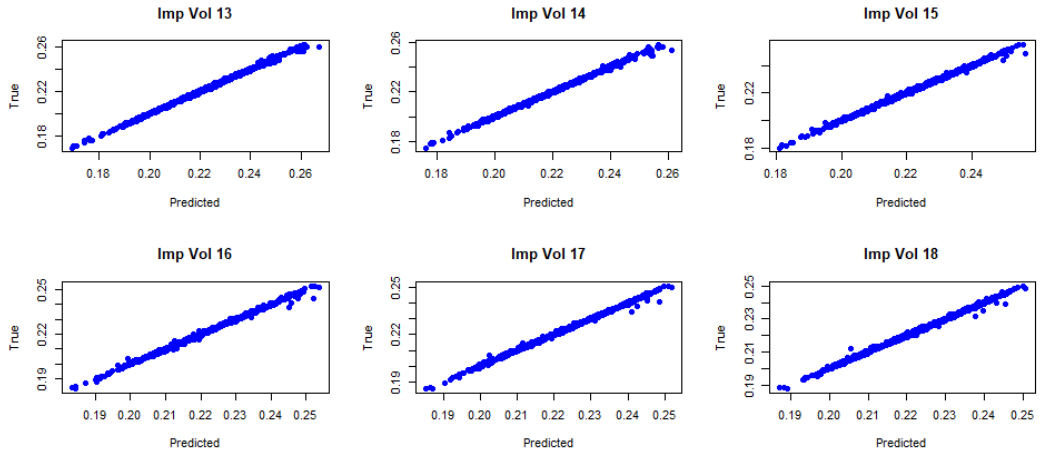

Fig. A.9: True vs Predicted (Heston) Implied Volatilities (13 to 18).

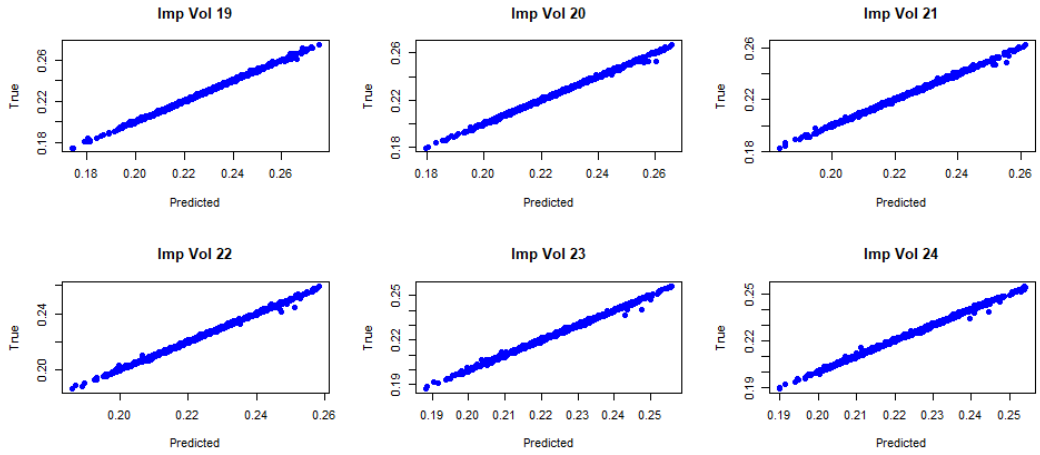

Fig. A.10: True vs Predicted (Heston) Implied Volatilities (19 to 24).

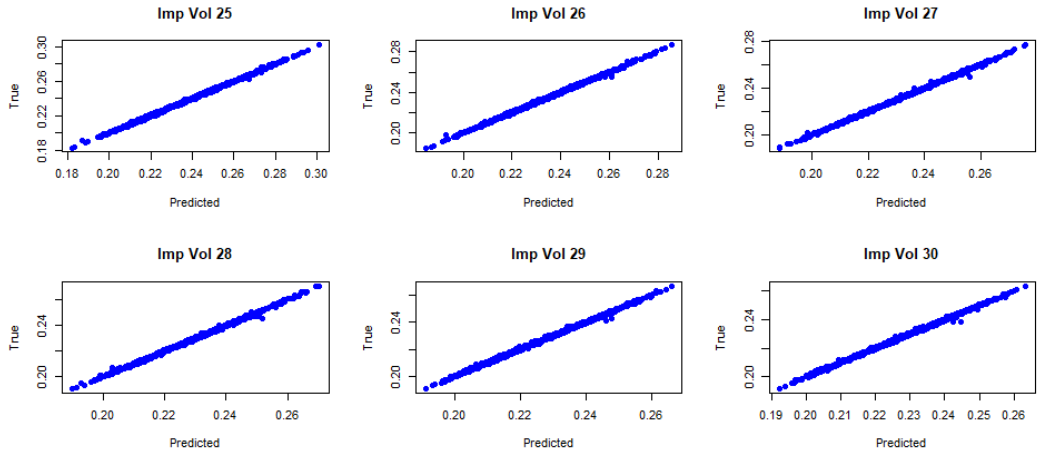

Fig. A.11: True vs Predicted (Heston) Implied Volatilities (25 to 30).

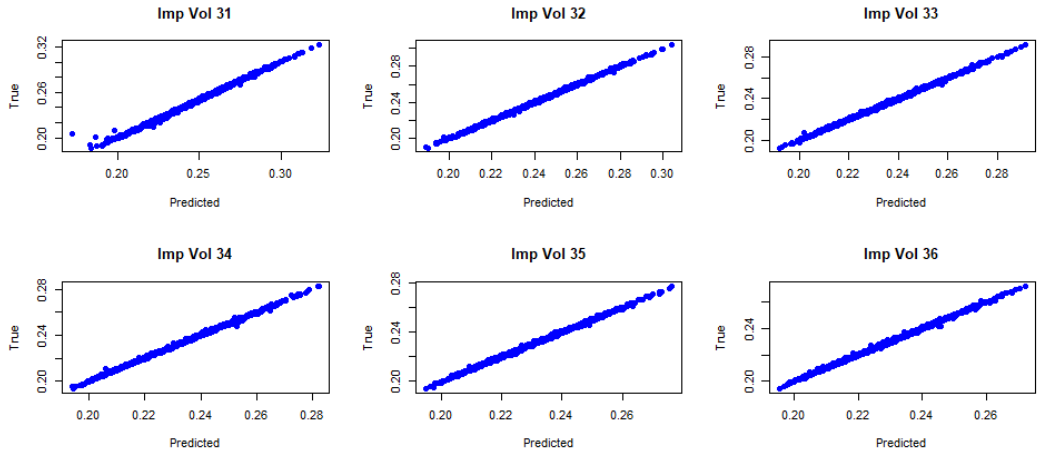

Fig. A.12: True vs Predicted (Heston) Implied Volatilities (31 to 36).

Appendix B

Indirect Method: Parameter Calibration

B.1 Indirect Method Parameter Calibration for SABR and Heston Model

The figures below display the true versus predicted parameter values for 1000 unseen samples under the indirect method for the SABR and Heston Model. This model calibration was conducted using one run through of the weighted least squares with the initial parameter values set equal to their means.

B.1.1 SABR Indirect Method Model Calibration

The figures illustrate the results obtained using the weighted least squares algorithm for the SABR model calibration.

Figure B.1 illustrates that the least squares algorithm was able to compute values of α relatively well. However, this algorithm failed to produce accurate values for ν and ρ. The histogram of absolute errors for each parameter illustrates the frequency of errors between the true and predicted parameters. Parameter α has a high frequency of errors between 0 and 0.05 and a very low frequency for other errors. Parameters ν and ρ have a slight decreasing trend in frequency of errors from 0 to 0.6. This further illustrates the poor performance of the least squares algorithm in predicting the parameters ν and ρ.

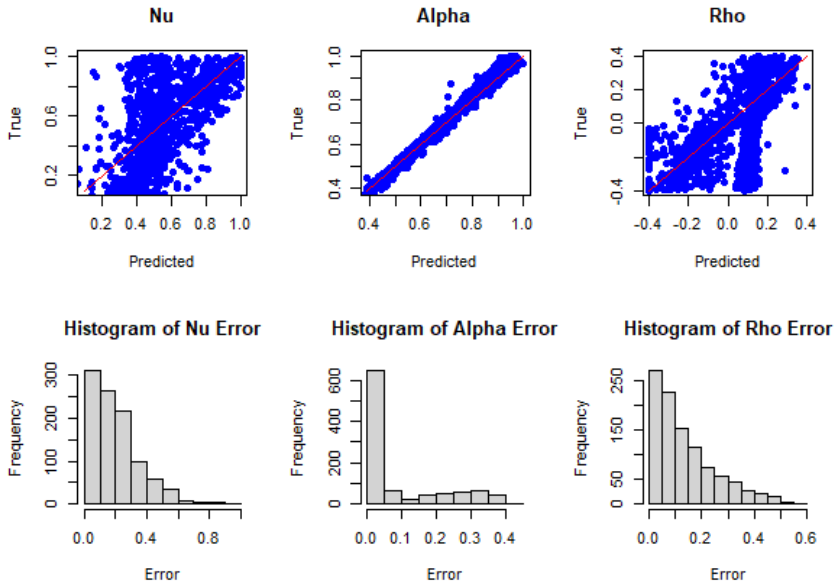

Fig. B.1: Indirect Method: True vs Predicted SABR Parameters.

B.1.2 Heston Indirect Method Model Calibration

The figures illustrate the results obtained using the weighted least squares algorithm for the Heston model calibration.

Figures B.2 and B.3 illustrate the relationship between the true parameter values and the predicted parameter values and histograms of their errors. Figure B.2 displays a linear relationship for parameter θ. However, the least squares algorithm was unable to predict ν_0 and κ accurately. This is further emphasised by the histograms. Parameter θ had a high frequency for a errors between 0 and 0.05, whereas the other two models exhibit a high frequency for larger errors. Figure B.3, suggests that both σ and ρ fail to capture the linear relationship between the true and the predicted. This is also evident from their error histograms. It is clear from the plots above that the least squares algorithm failed to predict parameter values similar to the true parameter values for ν_0, κ, σ and ρ. In addition, the least squares algorithm for model calibration was computationally expensive and heavily relied on the choice of initial parameters. Therefore, in order to accurately calibrate the Heston model parameters using this method, one would require more sophisticated methods of initial parameter selection which will in turn increase the computation time of the algorithm.

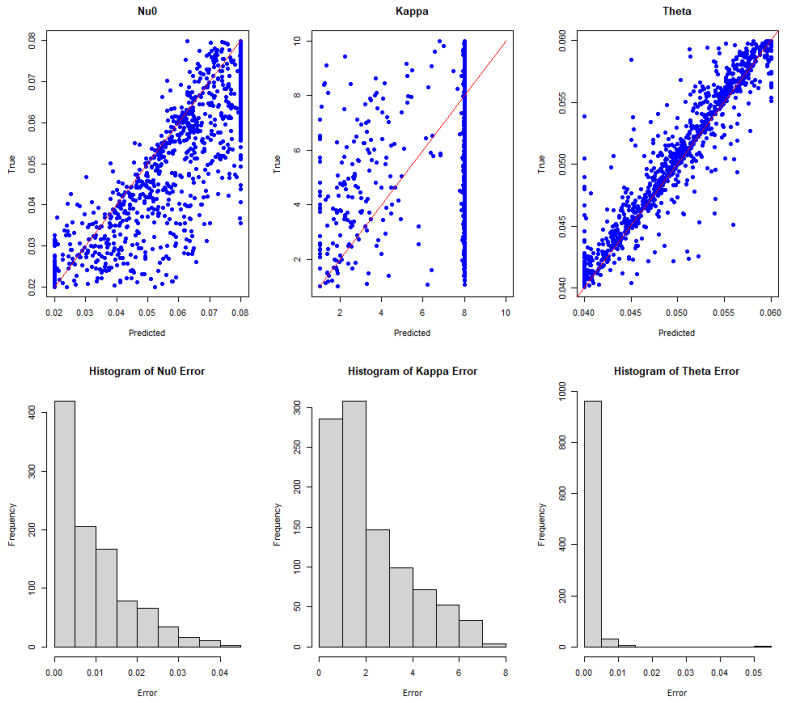

Fig. B.2: Indirect Method: True vs Predicted Heston Parameters ν_0, κ and θ .

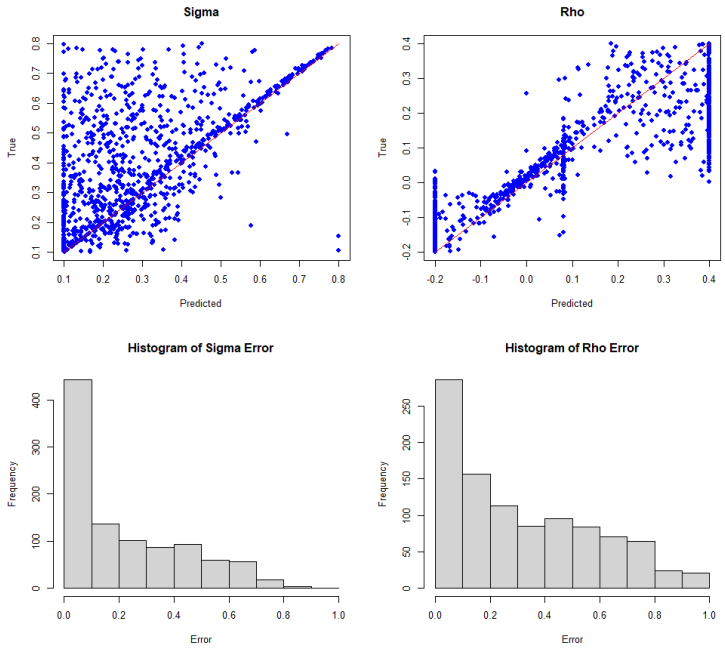

Fig. B.3: Indirect Method: True vs Predicted Heston Parameters σ and ρ .