# Heston model Calibration in the cryptocurrency market

Solal Danan - Luca Wang

June 17, 2024

# Contents

# 1 Introduction

Substantial progress has been made in option pricing since Black-Scholes-Merton was introduced. Most of the progress has tackled the problematic assumptions underlying the model, namely constant volatility and log-normally distributed stock prices. There is now abundant empirical evidence of persistent skewness and the presence of jumps in financial markets. This evidence has made it clear that making volatility stochastic and allowing for jumps can significantly improve the accuracy of option pricing. Strangely enough, BSM is so convenient to use that practitioners still use it despite its significant flaws. The model's influence on practitioners has been so extensive that derivatives traders quote options in terms of implied volatility, which is uniquely attributed to BSM. As Riccardo Rebonato famously stated:

*"Implied volatility is the wrong number you put in the wrong formula to get the right price."*

Nevertheless, alternative models to improve option pricing have been developed. These include local volatility models Dupire et al. (1994), stochastic volatility models Hagan, Kumar, Lesniewski, and Woodward (2002), Heston (1993), and stochastic volatility models with jumps, either in the stock price process Bates (1996) or in both stock price and volatility processes Duffie, Pan, and Singleton (2000). Many of these models have been proven to fit market prices quite well. The disadvantage is that some of these models either lack a closed-form solution or only have a quasi-closed form solution and can be computationally intensive. The rise of cryptocurrencies, a new asset class that initially disrupted financial markets but is slowly being adopted by institutional investors, is an interesting subject of research given the peculiar characteristics of how the underlying asset behaves. Given the significant volatility and frequency of jumps in the crypto market, an even more interesting question is how crypto options should be priced. In this project, we evaluate the in-sample performance of stochastic volatility models without jumps when pricing crypto derivatives, exploring alternative calibration methods based on neural networks. We begin by conducting a literature review of option pricing models. After establishing the theoretical framework necessary to understand the data analysis, we discuss the methodology adopted during our research, mainly focusing on the optimization techniques and neural network approaches used to calibrate the model parameters to market prices. Finally, we present and interpret the results of our analysis.

# 2 Literature Review

## 2.1 Overview of Option Pricing Theory

Classic options pricing theory began with Black and Scholes (1973) and Merton (1973), who provided the first analytical solution for pricing options by assuming stock prices follow a geometric Brownian motion with constant volatility. The Black-Scholes-Merton (BSM) model received immense attention and was widely adopted by traders due to its convenience and straightforward computation. Indeed, at that time, traders assumed that volatility was constant, as the BSM model proposed. After the Black Monday Crash in 1987, market participants started pricing out-of-the-money options at higher implied volatilities, giving rise to the so-called skew.

As discussed by Hull (2018), innovation in option pricing led to more advanced models, such as the family of stochastic volatility models. The nature of volatility makes it tempting to model not only stock prices but also volatility as a stochastic process. Following the work of Scott (1987), Hull and White (1987), and Stein and Stein (1991), Heston (1993) introduced the first stochastic volatility model with a closed-form solution. Heston's model captures the leverage effect, first documented by Black (1976), and the skew observed in the options market. A similar model (SABR) was introduced by Hagan et al. (2002). The difference between the two models is that Heston's model includes mean reversion, while the SABR model does not account for mean reversion in the volatility process. After the introduction of Heston's model, stochastic volatility models have been further improved by including jumps. The idea was initially introduced by Merton (1976). In particular, Bates (1996) and Scott (1997) account for stochastic volatility while allowing for jumps in the stock price process.

Further progress has been made by extending stochastic volatility models to allow for jumps in both stock price and volatility processes. Duffie et al. (2000) introduced a jump-diffusion model that models volatility with an affine process that can jump up suddenly. However, the model does not allow for volatility to jump down and return suddenly to normal levels.

Other than stochastic volatility models, the local volatility model has also been well-received by practitioners. The local volatility model was introduced by Dupire et al. (1994) and attempts to capture the instantaneous volatility of the underlying asset at any given point. In contrast, Black-Scholes uses the average of the instantaneous volatilities between the spot price and the strike price. Dupire's local volatility model is complete and consistent. "Complete" means that hedging involves only the underlying asset, and "consistent" means there are no contradictions. Dupire does not introduce additional stochastic processes other than the underlying asset's price. The local volatility model backs out the implied distribution for the asset price (which is not log-normal) using observed option prices across strikes. Indeed, the local volatility surface can be derived simply from the implied volatility surface using Dupire's formula. Due to its convenience, Dupire's local volatility model is widely used by investment banks in their day-to-day risk management. When comparing local volatility models and stochastic volatility models, Hagan et al. (2002) praised local volatility models for their precision in fitting market data but stated that they perform poorly when forecasting future movements of the volatility surface under new market conditions.

## 2.2 Cryptocurrency Markets and their Characteristics

Despite cryptocurrencies having been around for more than 15 years, institutional investors have only recently begun to adopt and recognize the asset class. In particular, the approval of the spot Bitcoin ETF in January 2024 and the more recent approval to list spot Ether ETFs by the SEC marked a substantial shift in regulators' and investors' sentiment toward the asset class. The approval of these ETFs paved the way for institutional investors to invest in the cryptocurrency market in a safe and regulated manner. Numerous papers, including Liu and Tsyvinski (2021) and Hu, Parlour, and Rajan (2019), have shown that cryptocurrencies such as Bitcoin and Ethereum exhibit unique characteristics, such as very high volatility and low correlation with traditional asset classes. Indeed, the unique risk-return profile of cryptocurrencies suggests they can be integrated into traditional investment portfolios for diversification purposes.

## 2.3 Application of Option Pricing Models in the Cryptocurrency Markets

While there is extensive research on the characteristics of cryptocurrencies, little work has been done on pricing cryptocurrency options. Madan, Reyners, and Schoutens (2019) compare the performance of several models (Black-Scholes, Laplace, Variance Gamma-related models, and Heston) when pricing BTC European options. They find that Heston and VG-CIR are the best-performing models. Hou, Wang, Chen, and Härdle (2020) price BTC options using the stochastic volatility model with correlated jumps (SVCJ) proposed by Duffie et al. (2000) and compare the results to a flexible co-jump model by Bandi and Reno (2016). They also compare how those models perform relative to the stochastic volatility model with jumps in the stock price process only Bates (1996). They find that adding jumps to the returns and volatility improves the fit of these models. Contrary to the leverage effect observed in the stock market, they observe that the correlation between return and volatility is significantly positive. Their interpretation is that since BTC price is not informative, as there is no fundamental price as in the equity market, BTC price is driven more by emotion and sentiment. Finally, they conclude that option prices are significantly affected by jumps in the returns and volatility processes.

## 2.4 Summary of Literature Review

Overall, research suggests that pricing cryptocurrency options is a more complicated and cumbersome task due to the frequent jumps in the price process. For this reason, the models that perform best are stochastic volatility models that account for jumps. While these models are more precise than simpler models that have closed-form or quasi-closed-form solutions, they are computationally intensive to calibrate and involve the use of methods such as the Fast Fourier Transform to solve for option prices. Due to the complexity of these models, in this project we focus on the Heston model and evaluate the different ways calibration can be conducted. The findings will apply to any similar model, such as the Bates model.

# 3 Theoretical Framework

## 3.1 Heston Model

The Heston's stochastic volatility model is specified as:

$$\frac{dS(t)}{S(t)} = \mu dt + \sqrt{V(t)}dW_1,$$
$$dV(t) = \kappa(\theta - V(t))dt + \sigma\sqrt{V(t)}dW_2. \quad dW_1 \cdot dW_2 = \rho dt$$

$V(t)$: initial variance, $\theta$: long run variance, $\kappa$: mean reversion, $\sigma$: volatility of volatility. $\rho$ is usually negative and accounts for the skew in the market. $V(t)$ is always positive if the Feller condition $2\kappa\theta \geq \sigma^2$ is satisfied. This condition is often violated in practice. Similarly to Black-Scholes-Merton, Heston is governed by the following (Garman) partial differential equation:

$$\frac{\partial C}{\partial t} + \frac{S^2V}{2}\frac{\partial^2 C}{\partial S^2} + (r-q)S\frac{\partial C}{\partial S} - (r-q)C + \kappa(\theta - V)\frac{\partial C}{\partial V} + \frac{\sigma^2 V}{2}\frac{\partial^2 C}{\partial V^2} + \rho\sigma SV\frac{\partial^3 C}{\partial S \partial V} = 0,$$

The pricing formula takes the form of Black-Scholes:

$$C(S_0, K, V_0, t, T) = SP_1 - Ke^{-(r-q)(T-t)}P_2,$$

Where $P_1$ is the delta of the European call option and $P_2$ is the conditional risk neutral probability that the asset price will be greater than K at the maturity. $P_1$ and $P_2$ are defined via the inverse Fourier transformation

$$P_j = \frac{1}{2} + \frac{1}{\pi}\int_0^\infty \mathrm{Re}\left[\frac{e^{-iu\ln K}\Phi_j(S_0, V_0, t, T, u)}{iu}\right]du, \quad j = 1, 2.$$

The characteristic functions $\phi_1$ and $\phi_2$ take the form

$$\phi(S_0, V_0, \tau; \phi) = \exp\left[C_j(\tau;\phi) + D_j(\tau;\phi)V_0 + i\phi S_0\right], \quad \tau = T - t,$$

$$C(\tau, \phi) = (r-q)\phi\tau + \frac{\kappa\theta}{\sigma^2}\left[(b_j - \rho\sigma\phi + d)\tau - 2\ln\left(\frac{1 - ge^{d\tau}}{1-g}\right)\right],$$
$$D(\tau;\phi) = \frac{b_j - \rho\sigma\phi + d}{\sigma^2}\left[1 - e^{d\tau}\right]\frac{1}{1 - ge^{d\tau}},$$

where

$$g = \frac{b_j - \rho\sigma\phi + d}{b_j - \rho\sigma\phi - d},$$
$$d = \sqrt{(\rho\sigma\phi - b_j)^2 - \sigma^2(2u_1\phi - \phi^2)},$$
$$u_1 = 0.5, \quad u_2 = -0.5,$$
$$a = \kappa\theta, \quad b_1 = \kappa + \lambda - \rho, \quad b_2 = \kappa + \lambda.$$

## 3.2 Calibration techniques

For the Heston model, we are looking to calibrate the set of parameters $\Theta = (\kappa, \theta, \sigma, \rho, V_0)$. We highlight that there are different methods to calibrate the model, as one can choose whether to estimate the parameters statistically (under the physical, or $P$, measure) or calibrate them to option prices (under the risk-neutral, or $Q$, measure). While $\sigma$, $\rho$, and $V_0$ are technically parameters under the $P$ measure, they are usually calibrated rather than estimated. This creates a discrepancy between the $P$ and $Q$ worlds. Commonly, when fitting the Heston model, $\kappa$ cannot be too large, otherwise, as practitioners say, it "kills the skew." We will see later that this does not really apply to crypto due to the presence of positive skewness. To calibrate the model we define the objective function:

$$f(\Theta) = \frac{1}{n}\sum_{i=1}^{n}(\text{marketprice}_i - \text{modelprice}_i)^2$$

As shown in Mikhailov and Nögel (2004), the appropriate choice of weight factors and the inclusion of a penalty factor are critical for accurate calibration. We introduce penalty factors to our objective function to see how the results change. However, we do not introduce weights in the objective function. The penalty factor we introduce is L2 regularization, commonly used in Ridge Regression. We define a cost function:

$$\text{Cost function} = \text{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2$$

where RSS is equal to the residual sum of squares, and the regularization term encourages smaller coefficients, which helps reduce model complexity and multicollinearity. In the results section, we show how increasing the parameter $\lambda$ changes calibration results. Optimization using regularization makes more sense since we are not only looking to fit the data but also looking to keep the model simple and generalizable to new data.

Once the objective function is defined, we choose the type of optimization scheme that minimizes the objective function. Optimization schemes can be local, global, or hybrid. Local optimization is the least computationally intensive and the fastest but can end up in a local minimum. In other words, they rely heavily on the initialization parameters. Global optimization is more precise and avoids ending up in a local minimum but converges rather slowly after finding the search space where the global minimum could be located. This suggests that the best optimization for accuracy and speed is a hybrid

scheme that combines global and local optimization schemes. Examples of global optimization schemes are Particle Swarm Optimization (PSO), Simulated Annealing, and Differential Evolution, while examples of local minimization schemes are the Levenberg-Marquardt Algorithm, Simplex, Conjugate-Gradient, and BFGS.

## 3.3 Optimization schemes

This section provides a brief explanation of the different optimization schemes used. The Levenberg-Marquardt algorithm combines gradient descent and Gauss-Newton methods. While not particularly effective for high-dimensional problems, it performs well when the solution is near optimal. Simplex is a derivative-free method, useful for noisy functions, relying on a geometrical approach to explore the search space. This makes it suitable for high-dimensional spaces but can be slower than other methods. Conjugate Gradient uses gradients to find the minimum of a function, making it suitable for large-scale problems where computing the Hessian matrix is impractical. Finally, BFGS is an iterative method that uses both gradient and an approximation of the Hessian. Considering the model's dimensionality and the aim for maximum accuracy, Levenberg-Marquardt appears to be the most suitable choice. Differential evolution (DE) is a global optimization algorithm that compares the attributes of randomly chosen individuals from a population. The algorithm continuously reshuffles a population of candidate solutions by integrating new candidate solutions and keeping only those that improve the optimization problem. Unlike many of the optimization schemes discussed earlier, differential evolution does not require the objective function to be differentiable.

## 3.4 Neural Network

In addition to using standard optimization schemes, we expand our analysis to explore how neural networks could be implemented in the calibration of a stochastic volatility model. Given the limited size of our dataset, we investigate the use of neural networks by augmenting our data. We apply synthetic data generation augmentation, which involves parametric perturbation. We scale our numerical features, as scaling helps the neural network converge faster and more efficiently by standardizing the range of the input data. Additionally, we one-hot encode the categorical feature `option_type` to help the neural network differentiate between put and call options. To train the model, we split the augmented dataset using an 80% train-test split. This split allows us to test our model on data different from that which it has been trained on, in order to analyze its generalization to new, unseen data.

### 3.4.1 Model Architecture

We implemented different neural network architectures in the calibration procedure, including a Long Short-Term Memory (LSTM) recurrent neural network to see if its properties would improve performance, and a more complex architecture. We focused on the latter, as it showed more promising results. The code for the model architecture can be found at the end of the appendix. The neural network model consists of several layers:

- **Dense Layer:** These layers are also called fully connected layers. In a dense layer, each neuron is connected to every neuron in the previous layer.
- **Batch Normalization:** This technique normalizes the input of each layer, which helps the network train faster and be more stable. It accomplishes this by scaling the activations to have a mean of 0 and a variance of 1.
- **Activation Function:** The ReLU (Rectified Linear Unit) activation function is defined as $f(x) = \max(0, x)$. It introduces non-linearity into the network, enabling it to learn more complex patterns. (We also tried the ELU activation function, but it yielded less interesting results). As our problem is a regression problem where we aim to predict numerical variables (the Heston parameters), the final layer does not use an activation function; only the weighted sum is calculated and returned.

- **Dropout:** This regularization technique randomly sets a fraction of input units to 0 at each update during training time, which helps prevent overfitting. For example, a dropout rate of 0.3 means 30% of the neurons will be set to 0 at each update.

### 3.4.2 Optimizer and Loss Function

The model is compiled with the Adam optimizer and the mean squared error loss function:

- **Adam Optimizer:** Adam (Adaptive Moment Estimation) is a stochastic optimization algorithm because it uses randomness to find an approximate solution by using subsets of data (batches) rather than the entire dataset at each iteration. Adam optimizer that can handle sparse gradients on noisy problems. It computes individual adaptive learning rates for different parameters. The update rule for Adam is defined as follows:

$$\theta_{t+1} = \theta_t - \frac{\alpha \cdot \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{1}$$

  where $\hat{m}_t$ and $\hat{v}_t$ are the estimates of the first moment (mean) and the second moment (uncentered variance) of the gradients, respectively.

- **Learning Rate:** The learning rate is set to 0.001, which controls how much the model is changed in response to the estimated error each time the model weights are updated. Finding the optimal learning rate depends on many factors and has been the subject of extensive research, so we will not delve into this aspect here.

- **Loss Function:** A custom loss function was used, based on the mean squared error (MSE), which measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value. We will provide more insights on this function later, as it played an important role in the calibration.

### 3.4.3 Number of Parameters

The number of parameters in a neural network is determined by the number of weights and biases in each layer. Here, we used Dense layers; to compute the number of parameters in a dense layer with $n$ input units and $m$ output units, there are $n \times m$ weights and an additional $m$ biases. Summing these across all layers gives the total number of parameters in the model.

The input_dim is seven and corresponds to the number of feature we used.

- First Dense Layer: $(7 \times 128) + 128 = 896 + 128 = 1024$
- First Batch Normalization: $4 \times 128 = 512$
- Second Dense Layer: $(128 \times 128) + 128 = 16512$
- Second Batch Normalization: $4 \times 128 = 512$
- Third Dense Layer: $(128 \times 64) + 64 = 8256$
- Third Batch Normalization: $4 \times 64 = 256$
- Fourth Dense Layer: $(64 \times 32) + 32 = 2080$
- Fourth Batch Normalization: $4 \times 32 = 128$
- Output Dense Layer: $(32 \times 5) + 5 = 165$

Therefore, the total number of parameters in the neural network is 1024 + 512 + 16512 + 512 + 8256 + 256 + 2080 + 128 + 165 = 29345.

# 4 Methodology

## 4.1 Data Collection

We source our data from Deribit, the unregulated crypto exchange that handles the highest volume and liquidity for BTC and ETH options. Although Deribit does not offer a formal database, the sourced data pertains to the orderbook informations on the exchange, collected on May 24, 2024. To filter out insignificant prices, we used only the depth one of the orderbook to have the best bid and ask prices. After setting up the API, we source the index price (BTC, ETH), strike price, implied volatility, maturity, and option type for each option. This data is then used to calibrate the Heston model. Some of the option prices derived from Deribit appear problematic, being either too low or too high. To resolve this, we implement a Black-Scholes-Merton pricer and re-price each option using all the other inputs provided by the exchange. For most options, the prices are very similar, while for obviously wrong values, we substitute them with the newly calculated price.

Furthermore, we eliminate any option that has a maturity below one day since 0DTE (zero-days-to-expiration) options could cause problems in the calibration procedure. For BTC, we obtained 578 options with maturities ranging from 1 day (May 25, 2024) to 307 days (March 28, 2025). Option strikes range from \$10,000 to \$200,000 USD, while BTC is trading at around \$68,800 USD. The implied volatilities range from 31% to 100%. For ETH, we obtained 502 options with maturities ranging from 1 day (May 25, 2024) to 307 days (March 28, 2025). Option strikes range from \$800 to \$20,000 USD, while ETH is trading at around \$3,730 USD. The implied volatilities range from 45% to 131%. For BTC, we further filter our data by selecting only strikes ranging from \$55,000 to \$80,000 and removing any missing values. We do the same for ETH by selecting only strikes from \$2,500 to \$6,000. We are left with 184 and 275 options for BTC and ETH, respectively.

## 4.2 Yield curve

We use the Treasury yield curve provided by the U.S. Department of the Treasury and perform interpolation for the missing maturities. We compared the interpolation results using Nelson-Siegel and cubic spline methods, and we chose to use cubic spline due to its smoother interpolation. This allows each of our options to be priced with a rate uniquely associated with its maturity. Since our options are mainly of short maturities ($< 1$ year), this step is crucial given the steep inversion we are currently facing at the front end of the curve. Results from the interpolation are shown in Figure 1. For maturities under one year, which we will utilize the most for pricing our options, the yield curve has a negative slope, and the spread between the 1-month yield and the 1-year yield is almost -60 basis points.

## 4.3 Calibration and Optimization Methods

The calibration of the model is conducted by implementing different optimization techniques. For the local minima, we use the Levenberg-Marquardt algorithm, Simplex, Conjugate Gradient, and BFGS. We also try to search for the global minimum using the differential evolution method. We later implement the differential evolution method with a penalty factor to see how the calibration process responds. We use the following set of initialization parameters: $\theta$=0.02, $\kappa$=0.2, $\sigma$=0.5, $\rho$=-0.75, V0=0.01. For local minima, the set of initialization parameters dramatically influences the final calibration results. To search for the global minimum, we introduce a set of bounds to reduce calibration time and exclude extreme values. For BTC, we implement the bounds $\theta$ (0, 1), $\kappa$ (0.01, 15), $\sigma$ (0.01, 6), $\rho$ (-1, 1), V0(0, 1). For ETH, we implement the bounds $\theta$ (0, 1.5), $\kappa$ (0.01, 15), $\sigma$ (0.01, 6), $\rho$ (-1, 1), V0(0, 1).

## 4.4 Performance Metrics

For practical purposes we select Mean Square Error (MSE) as the main performance metric. MSEs are shown in Table 1 and Table 3:

$$\text{mse} = \frac{1}{n}\sum_{i=1}^{n}(\text{marketprice}_i - \text{modelprice}_i)^2$$

$$\text{mse } \% = \frac{1}{n}\sum_{i=1}^{n}\frac{(\text{marketprice}_i - \text{modelprice}_i)^2}{\text{modelprice}_i^2 * 100}$$

The code includes other performance metrics such as Absolute Percentage Error (APE) and Average Relative Percentage Error (ARPE):

$$\text{ape} = \frac{\sum_{i=1}^{n}|\text{marketprice}_i - \text{modelprice}_i|}{\sum_{i=1}^{n}\text{marketprice}_i}$$

$$\text{arpe} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{|\text{marketprice}_i - \text{modelprice}_i|}{\text{marketprice}_i}\right)$$

## 4.5 Fit of Volatlity Surface

Stochastic volatility models are often calibrated on the implied volatility surface observed in the market. While calibrating on the volatility surface can often provide a better fit for the volatility dynamics, we pursue the route of calibrating on option prices. After calibrating the parameters for the Heston model, we calculate each option price using the Heston valuation function. This allows us to reverse-engineer the volatility value by plugging in the Heston option price and all the other inputs that are needed in the Black-Scholes-Merton model. As discussed in the literature review, implied volatility is a value that is uniquely associated with the Black-Scholes-Merton option pricing model. The Heston model does not really have an implied volatility. Nevertheless, this seemingly wrong approach from a theoretical standpoint allows us to compare how the Heston model fits the market without only looking at the price deviation.

## 4.6 Calibration using Neural Network

We calibrate the Heston model using a neural network. The network aims to predict all the required parameters of the stochastic volatility model: $\theta$, $\kappa$, $\sigma$, $\rho$, and $V_0$. To achieve this, we implemented, using the SciPy package, a function to calculate the Heston price using our dataset features as inputs and the parameters returned by the model. Using this characteristic function, we calculate the Heston prices of call and put options. The function code can be found in the appendix.

This `heston_option_price` function allows us to generate a predicted price. We then used the mean squared error (MSE) between this predicted price and the true price contained in our dataset as our loss function, which served as our target value. As explained, we used the ADAM optimizer to achieve more precise estimations. Because our dataset was augmented, we had many data points, so we trained the model using only a few epochs, where an epoch represents one complete pass through the entire training dataset. The training time of our calibration neural network did not exceed 15 minutes, making it an efficient approach for this problem.

# 5 Data Analysis and Results

## 5.1 Calibration Results

We first analyze the results for the calibration using the Levenberg-Marquardt algorithm. As shown in Table 1 and Table 3, the calibrated parameters for the Heston model using BTC options suggest a high degree of mean reversion ($\kappa > 8$). Practitioners often say that $\kappa$ cannot be too high, otherwise, it "kills the skew." However, as we can see from $\rho$, there exists a positive skew in the crypto market, which explains the high $\kappa$. The positive $\rho$ is a distinguishing feature of the crypto market. As mentioned in the literature review, Hou et al. (2020) attribute this result to the absence of fundamental pricing and the significant influence of emotion and sentiment in driving crypto prices. On the contrary, the equity market is affected by the so-called leverage effect, which can be empirically observed with a negative $\rho$. The long-term variance $\theta$ of 0.53 suggests that the Heston model estimated a long-term volatility of 73%. This is quite impressive compared to a standard annualized volatility of 16%. The estimated volatility of volatility $\sigma$ is 4.77, implying an extremely high level of kurtosis.

Results for ETH are shown in Table 2 and Table 4. We can immediately notice that volatility is higher, positive skew is weaker, and mean reversion is weaker for ETH compared to BTC. This is not surprising given the lower market capitalization of ETH compared to BTC, hence the higher volatility. We want to highlight that this analysis was conducted almost four months after BTC ETFs were introduced. By May 26th, the top 3 ETFs (GBTC, IBIT, FBTC) had surpassed $50 billion USD in AUM. As a consequence, the lower volatility of BTC can be partly attributed to the entrance of institutional investors into the BTC market. ETH will likely go through something similar after the recent approval of ETH ETFs by the SEC. Results obtained using other local minimization schemes, such as Simplex, Conjugate-Gradient, and BFGS, are mostly irrelevant as they either produce $\kappa = 0$ or a very negative $\rho$, which is contrary to what has been found in the literature. While the results from other minimization schemes have likely been influenced by the initialization parameters, the Levenberg-Marquardt algorithm seems to perform best, as results are very similar to the calibration conducted with differential evolution, a global minimization scheme.

Comparing the results from Levenberg-Marquardt and Differential Evolution in Table 1 and Table 3, we can see that they are almost identical. Things change when we introduce a penalty term factor that acts as a regularization term to prevent overfitting by penalizing large parameter values. After introducing a slight penalty, we notice that the mean reversion $\kappa$ goes down for both BTC and ETH, while the long-term variance $\theta$ increases. Other parameters remain almost untouched. The effect is accentuated when we increase the coefficient of the penalty factor. Indeed, $\theta$ goes from 0.54 to 0.70 for BTC and from 0.97 to 1.10 for ETH. Mean reversion $\kappa$ reaches as low as 4.40 for BTC and 2.60 for ETH. Using Levenberg-Marquardt, we obtain a mean square error (MSE) of 0.35% for BTC options, which suggests that the calibrated parameters fit the market prices extremely well. After introducing a penalty factor in the optimization problem and changing to a global optimization scheme, the mean square error (MSE) remains very low at 0.40%. For ETH, the fit is worse, as the mean square error (MSE) is slightly higher at 1.21%. In this case, the mean square error (MSE) does not change after introducing a penalty factor in the optimization problem and using Differential Evolution instead of Levenberg-Marquardt.

## 5.2 Volatility Surface

As shown in Figure 2 and Figure 3, the volatility surface for ETH options exhibits a much stronger positive skew compared to the surface for BTC options. This might be due to the recent approval of the ETH ETF by the SEC. Moreover, the volatility surface for ETH seems to be more strongly affected by term structure. To compare how the Heston model performed in fitting market prices, we plug Heston prices into the Black-Scholes-Merton model and retrieve the associated implied volatility for each option. Finally, we construct the volatility surface and compare it to the market one.

First, we look at how the two BTC volatility surfaces fit, as shown in Figure 4 and Figure 5. We immediately notice that the fit is pretty good except for the options that are immediately expiring and have high strikes. In this range, the Heston model struggles to replicate the high implied volatility for these options. This is expected with the Heston model, given that the literature has found that the model provides a decent fit at longer maturities but does not do a good job for shorter maturities.

Now, let's look at how the ETH volatility surface fits, as shown in Figure 6 and Figure 7. We notice that here the common area of divergence between the two surfaces is the area at high strikes, regardless of the maturity. For any option with strikes higher than 5000, the Heston model struggles to replicate the positive skewness observed in the market.

Finally, we calculate the average and maximum variation (implied volatility points) for the BTC and ETH volatility surfaces. For BTC, the average variation between the Black-Scholes-Merton and Heston surfaces is 1.4 volatility points, while the maximum variation is 8.8 volatility points. For ETH, we observe an average variation between the Black-Scholes-Merton and Heston surfaces of 1.6 volatility points, which is slightly higher than for BTC. The maximum variation at one point in the surface is observed at 8 volatility points, which is slightly lower than for BTC.

### 5.3 Neural Network results

To assess the accuracy of our calibration neural network and to maintain consistency with our previous analysis on the calibration of the Heston model, we used the following metrics:

- Average Mean Square Error (MSE) in (%),
- Average Mean Absolute Percentage Error (MAE) in (%),
- Average Relative Percentage Error (ARPE) in (%).

These averages are calculated by comparing the true prices to the prices generated using the output parameters from our calibration deep learning model. To avoid overfitting and to obtain a more accurate estimation of our model's performance, we computed these results on the predictions made on the test set—that is, new, unseen data. Additionally, to understand the efficacy of the deep neural network, we also assessed it using the $R^2$ indicator, which measures the proportion of variance in the dependent variable that is predictable from the independent variables.

Our most promising model had:

- Average Mean Square Error (MSE) in (%): 0.58,
- Average Mean Absolute Percentage Error (MAE) in (%): 6.45,
- Average Relative Percentage Error (ARPE) in (%): 0.06.

The ARPE is particularly low, likely due to the significant amount of data used, so it is perhaps not relevant.

As we refined and employed more advanced techniques for our neural network, the RMSE on the testing set decreased, and one interesting result is that our $R^2$ reached 0.95. This high $R^2$ value indicates that our model explains 95% of the variance in the target variable, demonstrating the model's strong predictive power.

# 6 Conclusion

## 6.1 Summary of Key Findings

Overall, results suggest that the crypto market is affected by a positive skew, which is contrary to what is observed in the equity market. Not surprisingly, the calibration of the Heston model returns a very high long-term volatility (104% for ETH and 83% for BTC). The stochastic volatility process in the Heston model is driven by a very high mean reversion. Moreover, the volatility of volatility is large, suggesting significant kurtosis and fat tails. Previous work focused only on BTC, but our project extended the analysis to ETH. We find that the volatility process for ETH is driven by a higher long-term volatility than BTC, while mean reversion is weaker. We also observe a worse fit in terms of mean square error (MSE). The implied volatility surface derived from the Heston model fits the market quite well, except for very out-of-the-money options and very short maturities. Local minimization schemes can differ greatly in results, and we find that Levenberg-Marquardt seems to be the one performing the best, as results are very similar to global optimization algorithms such as Differential Evolution. In calibrating the model, we highlight the importance of introducing boundary, weight, and penalty terms, as those factors can significantly affect the final parameter results. Indeed, we find that introducing a regularization term into the objective function lowers the mean reversion $\kappa$ and increases the long-term volatility $\theta$. The calibration using Neural Networks showed promising predictive power. Nevertheless, the robustness of this model needs to be further tested out-of-sample and using real (not augmented) data.

## 6.2 Limitations of the Study

In this project, we conducted our analysis only on the Heston model. The fit of the model could be improved by adding jumps, following the models proposed by Bates (1996) and Scott (1997). Further analysis could be conducted by allowing jumps in both the stock price and volatility process, as suggested by Duffie et al. (2000). Our project focused on the optimal way to calibrate stochastic volatility models; these results can be extended when calibrating other models. Regarding the use of neural networks, our project was limited by the lack of a large dataset. The use of augmented data does not perfectly resemble prices from the true stochastic process, and this introduces imperfections into

our analysis.

## 6.3 Recommendation for Future Research

Our analysis was limited to testing the fit of the model in-sample. We attempted to accommodate new data by implementing a penalty factor in the objective function. To test the robustness of the model, future research could be conducted on testing the calibration out-of-sample. Nevertheless, previous works and practitioners seem to agree that the Heston model must be recalibrated daily as parameters shift. Our project could also be extended by using more sophisticated neural networks with more parameters to capture nonlinear relationships. Although calibration using deep learning techniques showed less accurate results compared to traditional methods, we strongly believe that with the implementation of more advanced methods and increased focus on this approach, it could become a fast and accurate way to calibrate the Heston model for Bitcoin options.

# References


Federico M Bandi and Roberto Reno. Price and volatility co-jumps. *Journal of Financial Economics*, 119(1):107–146, 2016.

David S Bates. Jumps and stochastic volatility: Exchange rate processes implicit in deutsche mark options. *The Review of Financial Studies*, 9(1):69–107, 1996.

Fischer Black. Studies of stock market volatility changes. *Proceedings of the American Statistical Association, Business & Economic Statistics Section, 1976*, 1976.

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

Darrell Duffie, Jun Pan, and Kenneth Singleton. Transform analysis and asset pricing for affine jump-diffusions. *Econometrica*, 68(6):1343–1376, 2000.

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

Patrick S Hagan, Deep Kumar, Andrew S Lesniewski, and Diana E Woodward. Managing smile risk. *The Best of Wilmott*, 1:249–296, 2002.

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

Ai Jun Hou, Weining Wang, Cathy YH Chen, and Wolfgang Karl Härdle. Pricing cryptocurrency options. *Journal of Financial Econometrics*, 18(2):250–279, 2020.

Albert S Hu, Christine A Parlour, and Uday Rajan. Cryptocurrencies: Stylized facts on a new investible instrument. *Financial Management*, 48(4):1049–1068, 2019.

John Hull and Alan White. The pricing of options on assets with stochastic volatilities. *The journal of finance*, 42(2):281–300, 1987.

John C Hull. Options, futures, and other derivatives. global edition. *Harlow: United Kingdom: Pearson*, 2018.

Yukun Liu and Aleh Tsyvinski. Risks and returns of cryptocurrency. *The Review of Financial Studies*, 34(6):2689–2727, 2021.

Dilip B Madan, Sofie Reyners, and Wim Schoutens. Advanced model calibration on bitcoin options. *Digital Finance*, 1(1):117–137, 2019.

Robert C Merton. Theory of rational option pricing. *The Bell Journal of economics and management science*, pages 141–183, 1973.

Robert C Merton. Option pricing when underlying stock returns are discontinuous. *Jour-*

*nal of financial economics*, 3(1-2):125–144, 1976.

Sergei Mikhailov and Ulrich Nögel. *Heston's stochastic volatility model: Implementation, calibration and some extensions*. John Wiley and Sons, 2004.

Louis O Scott. Option pricing when the variance changes randomly: Theory, estimation, and an application. *Journal of Financial and Quantitative analysis*, 22(4):419–438, 1987.

Louis O Scott. Pricing stock options in a jump-diffusion model with stochastic volatility and interest rates: Applications of fourier inversion methods. *Mathematical Finance*, 7(4):413–426, 1997.

Elias M Stein and Jeremy C Stein. Stock price distributions with stochastic volatility: an analytic approach. *The review of financial studies*, 4(4):727–752, 1991.

# Appendix

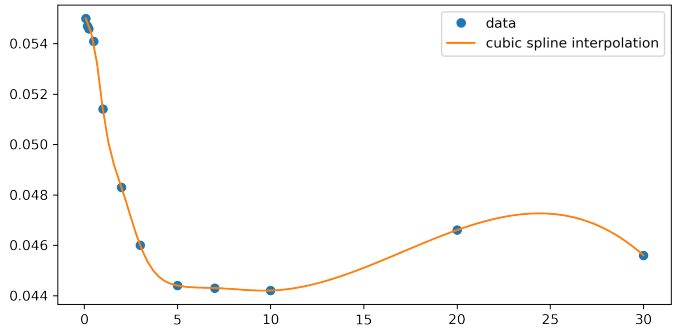


Figure 1: Implied Volatility surface for BTC options

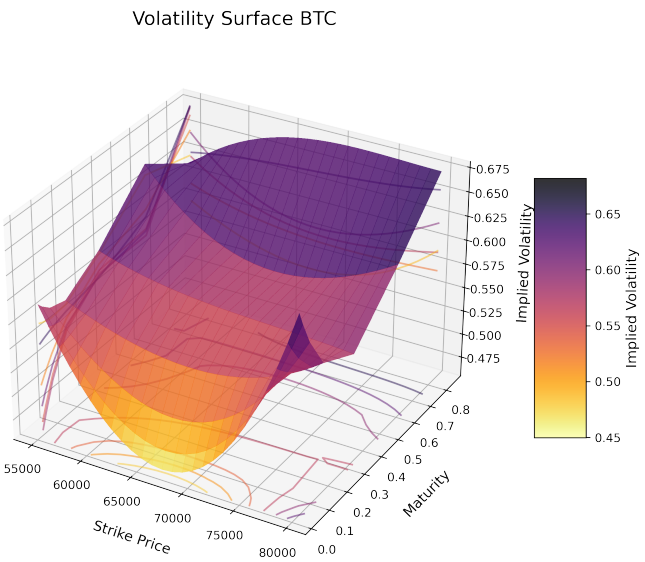


Figure 2: Implied Volatility surface for BTC options

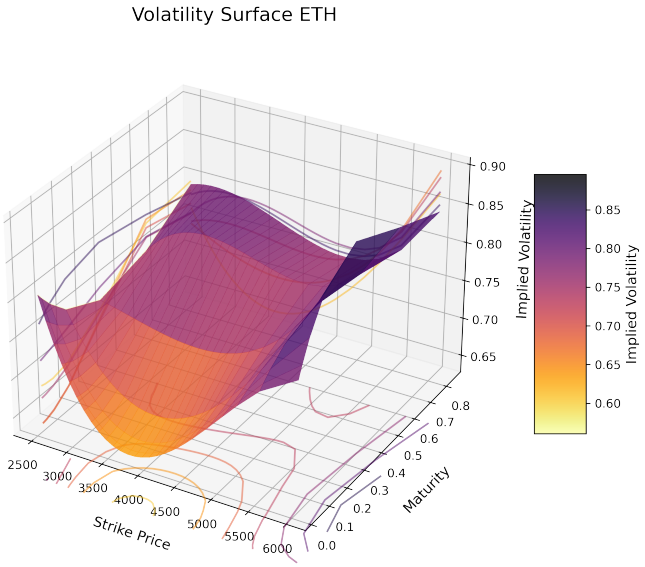


Figure 3: Implied Volatility surface for ETH options

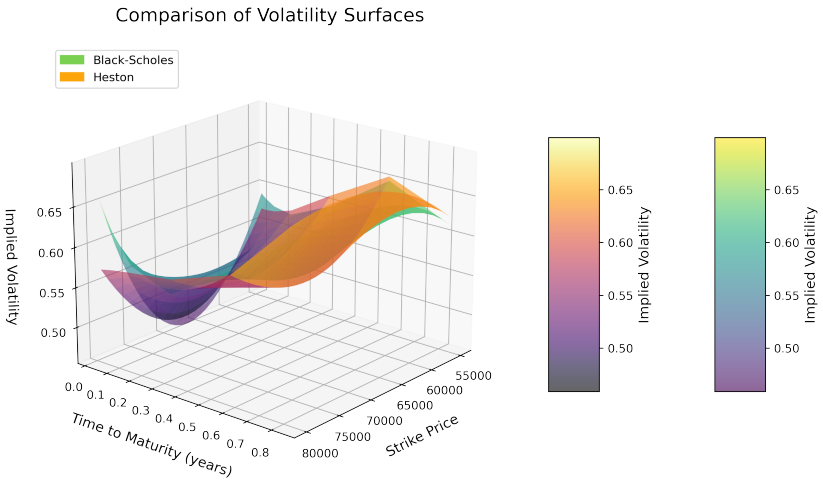


Figure 4: Volatility surface fit Heston vs Market implied (BTC)

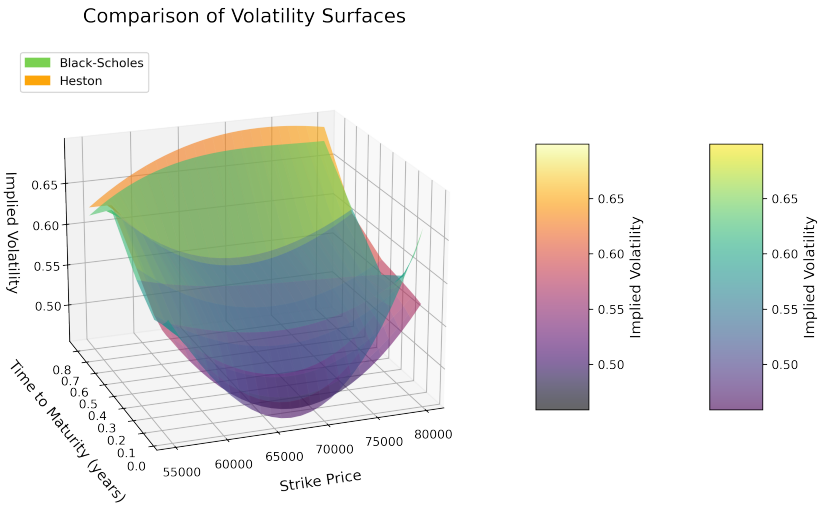


Figure 5: Volatility surface fit Heston vs Market implied (BTC)

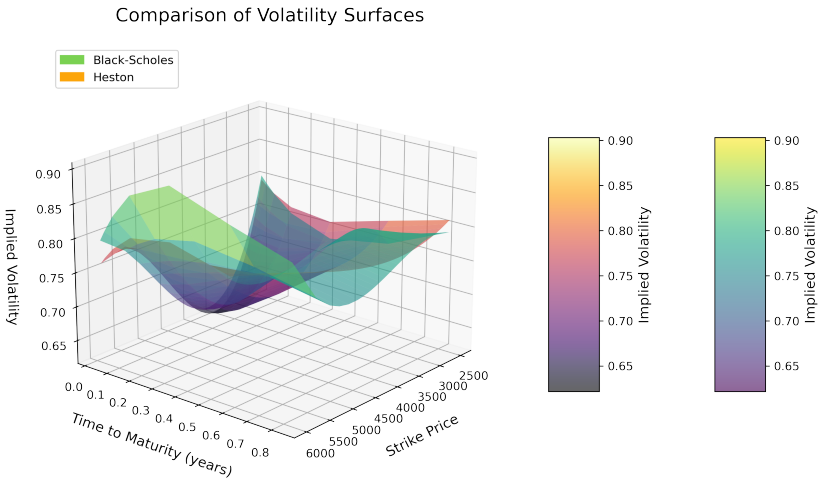


Figure 6: Volatility surface fit Heston vs Market implied (ETH)

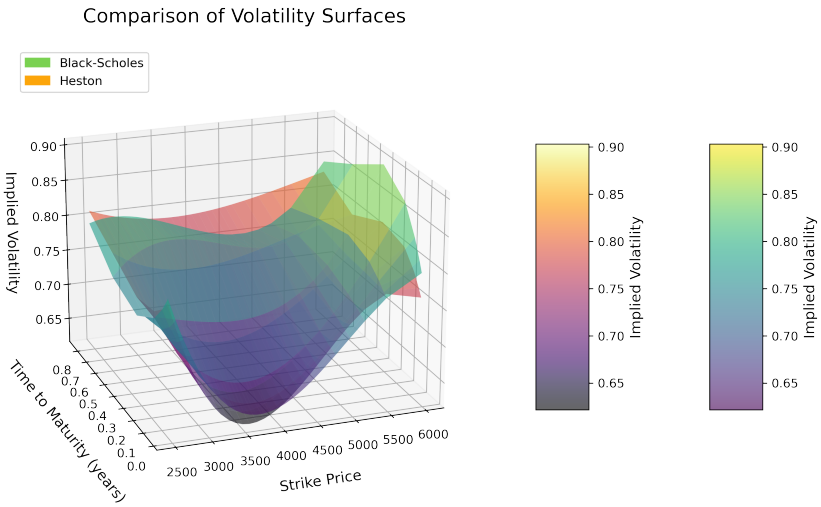


Figure 7: Volatility surface fit Heston vs Market implied (ETH)

Table 1: Calibration Results Using Various Methods for Heston (BTC)

| Method | MSE (%) | $\theta$ | $\kappa$ | $\sigma$ | $\rho$ | V0 |
|---|---|---|---|---|---|---|
| Levenberg-Marquardt | 0.35175 | 0.54365 | 8.81325 | 4.76334 | 0.10751 | 0.20366 |
| Simplex | 1.77276 | 0.34873 | 76.12230 | 0.56826 | -1.00000 | 0.03631 |
| Conjugate-Gradient | 3.17585 | 0.19445 | 0.00000 | 0.19519 | -0.64366 | 0.28359 |
| BFGS | 3.16860 | 0.20241 | 0.00000 | 0.17570 | -0.61869 | 0.28343 |
| DE (without penalty) | 0.35175 | 0.54427 | 8.78136 | 4.75813 | 0.10745 | 0.20376 |
| DE (with penalty) | 0.35950 | 0.59160 | 6.83486 | 4.41801 | 0.10312 | 0.20990 |
| DE (with high penalty) | 0.40196 | 0.70146 | 4.40616 | 3.91939 | 0.09401 | 0.21846 |

Table 2: Calibration Results Using Various Methods for Heston (ETH)

| Method | MSE (%) | $\theta$ | $\kappa$ | $\sigma$ | $\rho$ | V0 |
|---|---|---|---|---|---|---|
| Levenberg-Marquardt | 1.20620 | 0.97045 | 3.46455 | 4.34841 | 0.06925 | 0.37622 |
| Simplex | 1.20620 | 0.97051 | 3.46410 | 4.34836 | 0.06925 | 0.37622 |
| Conjugate-Gradient | 76.21939 | 0.05164 | 0.20156 | 0.49994 | -0.74951 | 0.79503 |
| BFGS | 4.15254 | 0.17334 | 0.00000 | 0.07475 | -0.52397 | 0.51049 |
| DE (without penalty) | 1.20620 | 0.97042 | 3.46475 | 4.34844 | 0.06925 | 0.37622 |
| DE (with penalty) | 1.20653 | 1.00140 | 3.22688 | 4.30992 | 0.06863 | 0.37763 |
| DE (with high penalty) | 1.21082 | 1.10777 | 2.60847 | 4.20140 | 0.06677 | 0.38151 |

Table 3: Results for selected BTC Options

| Strike | Market Value | Model Value | MSE Error (%) |
|---|---|---|---|
| 58000 | 134.29 | 122.81 | 0.7317 |
| 58000 | 134.29 | 122.81 | 0.7317 |
| 60000 | 226.38 | 222.60 | 0.0279 |
| 60000 | 226.38 | 222.60 | 0.0279 |
| 61000 | 290.26 | 297.21 | 0.0574 |
| 61000 | 290.26 | 297.21 | 0.0574 |
| 62000 | 389.22 | 394.81 | 0.0207 |
| 62000 | 389.22 | 394.81 | 0.0207 |
| 63000 | 531.13 | 521.88 | 0.0304 |
| 63000 | 531.13 | 521.88 | 0.0304 |
| 64000 | 666.27 | 686.44 | 0.0917 |
| 65000 | 896.35 | 898.13 | 0.0004 |
| 65000 | 896.35 | 898.13 | 0.0004 |
| 66000 | 1152.25 | 1167.78 | 0.0182 |
| 66000 | 1152.25 | 1167.78 | 0.0182 |
| 67000 | 1493.66 | 1506.25 | 0.0071 |
| 67000 | 1493.66 | 1506.25 | 0.0071 |
| 68000 | 1879.08 | 1922.35 | 0.0530 |
| 68000 | 1879.08 | 1922.35 | 0.0530 |
| 69000 | 2325.62 | 2349.68 | 0.0107 |
| 69000 | 2325.62 | 2349.68 | 0.0107 |
| 70000 | 1875.47 | 1929.69 | 0.0836 |
| 70000 | 1875.47 | 1929.69 | 0.0836 |
| 75000 | 15777.84 | 15643.72 | 0.0072 |
| 75000 | 15777.84 | 15643.72 | 0.0072 |
| 80000 | 14270.60 | 14126.10 | 0.0103 |
| 80000 | 14270.60 | 14126.10 | 0.0103 |
| Average MSE (%): 0.350 | | | |

Table 4: Results for selected ETH Options

| Strike | Market Value | Model Value | MSE Error (%) |
|---|---|---|---|
| 2700.00 | 1.45172 | 1.52200 | 0.2344 |
| 2700.00 | 1.45172 | 1.52200 | 0.2344 |
| 2800.00 | 3.12163 | 2.71612 | 1.6874 |
| 2800.00 | 3.12163 | 2.71612 | 1.6874 |
| 2900.00 | 6.17127 | 4.71108 | 5.5985 |
| 2900.00 | 6.18068 | 4.71108 | 5.6536 |
| 3000.00 | 11.37347 | 7.95251 | 9.0471 |
| 3000.00 | 11.37347 | 7.95251 | 9.0471 |
| 3100.00 | 16.59605 | 13.07922 | 4.4905 |
| 3100.00 | 16.58646 | 13.07922 | 4.4712 |
| 3200.00 | 24.01944 | 20.97668 | 1.6048 |
| 3300.00 | 37.77364 | 32.82664 | 1.7152 |
| 3300.00 | 37.77364 | 32.82664 | 1.7152 |
| 3400.00 | 55.53605 | 50.13064 | 0.9473 |
| 3400.00 | 55.53605 | 50.13064 | 0.9473 |
| 3500.00 | 83.15462 | 74.65432 | 1.0450 |
| 3600.00 | 119.28241 | 108.19604 | 0.8638 |
| 3600.00 | 119.28241 | 108.19604 | 0.8638 |
| 3700.00 | 166.07241 | 152.10959 | 0.7069 |
| 3700.00 | 166.07241 | 152.10959 | 0.7069 |
| 3800.00 | 128.24673 | 141.77832 | 1.1133 |
| 3800.00 | 128.24673 | 141.77832 | 1.1133 |
| 3900.00 | 94.82265 | 106.56621 | 1.5338 |
| 3900.00 | 94.82265 | 106.56621 | 1.5338 |
| 4000.00 | 69.72353 | 79.75133 | 2.0685 |
| 4100.00 | 51.22154 | 59.60081 | 2.6761 |
| 4100.00 | 51.15871 | 59.60081 | 2.7231 |
| 4200.00 | 39.15319 | 44.55571 | 1.9040 |
| Average MSE (%): 1.191 | | | |

Table 5: Selected Calibration Results using Neural Network (BTC)

| | True Price | Predicted Price | Error Percentage (%) |
|---|---|---|---|
| 311 | 12363.121180 | 11958.227539 | 3.275012 |
| 159 | 8929.889156 | 9226.750977 | 3.324362 |
| 433 | 11233.020318 | 12266.680664 | 9.201981 |
| 319 | 8859.566310 | 9271.804688 | 4.653031 |
| 504 | 16411.229944 | 15544.561523 | 5.280947 |
| ... | ... | ... | ... |
| 360 | 10179.868537 | 10707.724609 | 5.185294 |
| 422 | 18000.916770 | 17269.623047 | 4.062536 |
| 436 | 11959.052776 | 12147.405273 | 1.574978 |
| 554 | 20634.866884 | 18442.365234 | 10.625228 |
| 161 | 10909.096003 | 10805.225586 | 0.952145 |

# Code

Listing 1: Neural network model for Heston parameter calibration

```
def heston_calibration_neural_network(input_dim):
    model = Sequential([
        Dense(128, input_shape=(input_dim,)),
        BatchNormalization(),
        Activation('relu'),
        Dropout(0.3),

        Dense(128),
        BatchNormalization(),
        Activation('relu'),
        Dropout(0.3),

        Dense(64),
        BatchNormalization(),
        Activation('relu'),
        Dropout(0.2),

        Dense(32),
        BatchNormalization(),
        Activation('relu'),
        Dropout(0.1),
        Dense(5)  # Outputs five Heston parameters: kappa, theta, sigma, rho, v0
    ])
    model.compile(optimizer=legacy_optimizers.
    Adam(learning_rate=0.00), loss='mean_squared_error')
    return model
```

Listing 2: Function to compute call and put option prices using the Heston model

```
def heston_characteristic_function(u, S0, K, r, T, kappa, theta, sigma, rho, v0):
    xi = kappa - rho * sigma * 1j * u
    d = np.sqrt((rho * sigma * 1j * u - xi)**2 + sigma**2 * (-u * 1j - u**2))
    g = (xi - d) / (xi + d)
    C = r * 1j * u * T + (kappa * theta) / sigma**2 *
    ((xi - d) * T - 2 * np.log((1 - g * np.exp(-d * T)) / (1 - g)))
    D = (xi - d) / sigma**2 * ((1 - np.exp(-d * T)) / (1 - g * np.exp(-d * T)))
    return np.exp(C + D * v0 + 1j * u * np.log(S0))

def heston_option_price(S0, K, r, T, kappa, theta, sigma, rho, v0, option_type):
    if option_type == 1:  # Call option
        integrand = lambda u: np.real(np.exp(-1j * u * np.log(K)) / (1j * u) *
        heston_characteristic_function
        (u - 1j, S0, K, r, T, kappa, theta, sigma, rho, v0))
        integral, _ = quad(integrand, 0, np.inf)
        return np.exp(-r * T) * (S0 * 0.5 - np.exp(-r * T) / np.pi * integral)
    else:  # Put option
        integrand = lambda u: np.real(np.exp(-1j * u * np.log(K)) / (1j * u) *
        heston_characteristic_function
        (u - 1j, S0, K, r, T, kappa, theta, sigma, rho, v0))
        integral, _ = quad(integrand, 0, np.inf)
        return np.exp(-r * T) / np.pi * integral - S0 + K * np.exp(-r * T)
```