Modelling and Calibration Techniques For Fractional Black Scholes Model

Athul AR[1], Richard Obonyo[2]

[1]*Worldquant University*
[2]*Worldquant University*
[1]athular7@gmail.com , [2]richardobonyo12@gmail.com

Abstract

The Black Scholes model, despite being the dominant derivative pricing model in the financial analyst's toolkit, has shortcomings in numerous aspects. Fractional alternatives, based on the Fractal Market Hypothesis have been gaining foothold. The main difference between the standard Black Scholes model and Fractional Black Scholes model is that the latter model, long term memory of the time series process which is depicted by the hurst exponent of the time series data is used in pricing the option. The accuracy of pricing of options in the Fractal Black Scholes universe therefore depends on the accurate prediction of model parameters such as Hurst exponent and volatility. We leverage the power of statistical models and neural networks in the calibration and prediction of these model parameters. With these estimated parameters from the stock price data we are able to show that the Fractional Black Scholes model is much more accurate in pricing options as compared to the standard Balck Scholes model.

Keywords: *Fractional Brownian Motion, Neural Networks, Invertibility, Calibration*

1. Introduction

1.1 Problem Statement

The most widely used approach to model price movement is the Black Scholes model. Although quite easy to use, the model has been determined to be inadequate in predicting extreme market variations due to its numerous simplifying assumptions and choices. The Black Scholes model failed to predict black swan events like the economic crisis of 2008. Despite these shortfalls, the Black Scholes model is widely used as the primary modelling tool in finance, notably for pricing financial derivatives.

Benoit Mandelbrot, a French mathematician, noticed that option prices were not normally distributed and did not follow a Gaussian distribution as assumed by Black and Scholes but actually reflected access kurtosis with flat tail tendencies and the price distributions exhibited both long and short memory of price trends. Price movements assume α-stable and fractional brownian motion distributions with memory and could therefore be used to better predict extreme variations and movements of prices in the market.

The effectiveness of the fractional Black Scholes model based on Mandelbrot's theory requires accurately determining model parameters like volatility and Hurst exponent; which indicates the memory trends. These parameters are presently not well modelled as it is determined from time series data of financial instruments and this affects the predictive accuracy of the model. Correctly modelling and determining these parameters will allow the better calibration of the model which will improve price prediction, reduce losses incurred from wrongful pricing of derivatives, enable better anticipation and

prediction of market crises and black swan events and help generate profitable trading strategies based on financial derivatives.

1.2 Goals and Objectives

1.2.1 Goal

To study existing estimation and calibration techniques, design and implement tractable approaches for calibration of heuristic fractal models.

1.2.2 Objectives

- To study and review existing literature on fractal based price modelling methods for pricing of options.
- To design techniques and methods to help us better understand and model price movements in pricing models based on fractal Brownian motion as proposed by Mandelbrot and other researchers.
- To study the parameter(s) of the fractal brownian motion based pricing models and determine how to better predict the parameter(s) using models like Artificial neural networks, linear models and machine learning.
- To test the effectiveness and reliability of the designed model in predicting option prices by comparing our predictions with real market option price data sets.
- To compare, validate and calibrate the fractal Brownian motion pricing model with the Black Scholes pricing model and market prices so as to determine and improve the effectiveness in modeling option prices.

1.3 Literature Review

1.3.1 Fractional Black Scholes Model And Pricing Options Under The Fractional Brownian Motion

The dominant economic theory in the field of financial modelling is Black Scholes model backed up by ideas like Efficient Market Hypothesis. It was based on numerous simplifying but restrictive assumptions like efficient markets, perfect liquidity, rational agents, normal price distribution etc. Fractals, a brainchild of Benoit Mandelbrot came as a radical alternative, relaxing many erroneous assumptions, and using fundamental principles of behavioral economics. In particular, this approach allowed non-efficient markets, illiquidity, irrational agents, and general price distributions among numerous others.[1] These advances were frowned upon by academia for a long period due to loss of properties like finite volatility, but have found its way back to the mainstream lately.

The Fractional Black Scholes model is based on fractional Brownian motion which was originally introduced by Kolmogorov in 1940. Fractional Brownian motion is also known as a stochastic two sided Brownian motion process or generalized Gaussian processes defined mathematical as

$B_t^H,\ t \in R^+$ and $H \in [0, 1]$ is the Hurst exponent.

Consider the fractal differential equation

$$dS_t = \mu S_t dt + \sigma S_t dB_t^H \quad \rightarrow (1)$$

where, under the risk neutral measure

$$dS_t = rS_t dt + \sigma S_t dB_t^H \quad \rightarrow (2)$$

Equation (1) can be solved using the fractional Ito calculus to obtain our stock price propagation based on the fractional Brownian motion as below:

$$S_t = S_0 exp\left(\sigma B_t^H + \mu t - \tfrac{1}{2}\sigma^2 t^{2H}\right) \quad \rightarrow (3)$$

where S_0 is the current/initial stock price and r is the risk free interest rate

Hu and Oksendal[11,12] were able to improve the classical Black Scholes model by using Fractal Ito calculus and a fractal Geometric Brownian motion. We are able to derive the following closed form solution for a European call option.

$$\pi_c = S_t \phi(d_1) - Ke^{-r(T-t)}\phi(d_2) \quad \rightarrow (4)$$

$$d_1 = \frac{\ln ln\left(\frac{S_t}{k}\right) + r(T-t) + \frac{\sigma^2}{2}(T^{2H} - t^{2H})}{\sigma\sqrt{(T^{2H} - t^{2H})}}$$

$$d_2 = \frac{\ln ln\left(\frac{S_t}{k}\right) + r(T-t) - \frac{\sigma^2}{2}(T^{2H} - t^{2H})}{\sigma\sqrt{(T^{2H} - t^{2H})}} \quad \rightarrow (5)$$

Where t is the start period, T is the end period, S_t is the initial price at the start of the period, K is the strike price, ϕ is the cumulative standard normal distribution, r is the risk free rate, and H is the Hurst exponent.[11]

Put options can be priced using the put-call parity.

$$\pi_c - \pi_p = Ke^{-r(T-t)} \quad \rightarrow (6)$$

Hu and Oksendal[11,12] in their papers particularly use the concept of Wick product to address the no arbitrage issues of the fractional Black scholes model in their derivations of the pricing model. Detailed mathematical derivations and proofs of consideration of no arbitrage can be viewed in their papers.

Therefore based on equations 1 to 5 if we are able to model/determine the Hurst exponent and volatility as accurately as possible then we can accurately price our options based on the fractal Black Scholes model.

Traditionally, the idea of implied volatility is borrowed from Black Scholes model to compute volatility and Hurst exponent is estimated separately. We will therefore look at various methods for estimating the Hurst exponent and see how to improve them so that we can have more accurate pricing of European options. We will look at the rescale range (R/S) analysis method and the minimum covered area (Box counting) method of estimation of fractal index.

1.3.2 Rescale Range (R/S) Analysis Method

This approach for estimating Hurst exponent was developed by Mandelbrot and is a very widely used method to determine Hurst exponent.

The process for estimating the Hurst exponent using R/S analysis is as below.

1. Obtain the logarithm of the ratio of the stock price from the i^{th} period to the $(i-1)^{th}$ period where *for i= 1,2,3,……,n.* This helps to remove trends from our data.

$$X_t = log(\frac{X_i}{X_{i-1}}) \quad \rightarrow(7)$$

2. Compute the standard deviation on the sample size n of our data $s(n)$

3. We compute the quantity

$$\left(\frac{R}{S}\right)n = \frac{1}{S(n)}\left[sup\left(X_t - X_t^*\right) - inf\left(X_t - X_t^*\right)\right] \quad \rightarrow(8)$$

Where $X_t^* = \frac{1}{n}\sum_{i=1}^{n} X_i$ is the sample mean. $\rightarrow(9)$

4.We repeat 2 and 3 for various values of n and plot a graph of $log((\frac{R}{S})_n)$ against $log(n)$

5.Determine the slope of the line of best fit and that is the Hurst exponent for our data which can then be used in the fractal black Scholes model to price our options.

Please note that the Hurst exponent can also be more easily obtained by using the Ordinary least squares (OLS) method. [1]

1.3.3 Box Counting Method For Estimating Fractal Dimension

The approach in this section was first suggested in 1919 by Hausdorff it basically involves plotting the time series data on a graph and the surface of the time series data on the graph is divided into small cells(boxes) of size h the number of boxes $N(h)$ is then counted[8][13].

Note that; $h = t_i - t_{i-1} = \frac{b-a}{m}$ $\rightarrow(10)$

where $i = 1, 2, 3, ..., m$ and $N(h) \sim \left(\frac{1}{h}\right)^D$ $\rightarrow(11)$

The computation of $N(h)$ is repeated for various values of h and a graph of $log(N(h))$ against $log(h)$ is plotted with the slope being the fractal dimension D. The hurst exponent H is then determined from $H = 2 - D$. It is to be noted that OLS estimation can be applied to obtain the fractal dimension here as well.[13].

1.3.4 Computing Volatility And Drift Of Time Series Data In A Fractal Black Scholes Model

Volatility

Using the log returns (R_t , *t= 1,2,3,……,n*) for the time series data, we compute average return;

$$R_t^* = \frac{1}{n}\sum_{i=1}^{n} R_i \quad \rightarrow(12)$$

and sample variance is computed;

$$S_t^* = \frac{1}{n-1} \sum_{i=1}^{n} (R_t^* - R_i)^2 \quad \rightarrow (13)$$

the time series volatility is then computed as below;

$$\sigma = \sqrt{\frac{S_t^*}{(dt)^{2H}}} \quad \rightarrow (14)$$

and the drift of the time series is compute as well;

$$\mu = \frac{R_t^*}{dt} + \frac{\sigma^2}{2} \quad \rightarrow (15)$$

The volatility and drift can then be used to predict future time series and in option pricing[15].

1.3.5 Calibration: Machine Learning and Artificial Neural Network Approaches To Option Pricing

Model calibration is a complementary problem of option pricing, since any pricer involves parameters based on theoretical models, which needs to be decided prior. There has been a lot of advances in financial model calibration techniques. However, many of them were, understandably, to improve Black Scholes model. The initial techniques included the calibration of the implied volatility complying with the assumption of constant volatility. But these were inadequate primarily due to the non-constant relationship of the volatility with the option term.[2] Many subsequent approaches tried to account for this mismatch by improving the model. The notable ones include smile-consistent pricing, stochastic volatility models, local volatility models, jump diffusion models etc.

However, relatively fewer advances have been made for models set in the Mandelbrotian world. The noteworthy ones are Mare et. al[5] and Li and Chen[6] These have focused on exploiting the theoretical and heuristic advantages that fractal market hypothesis has over the efficient market counterpart. The results are arguably superior to the vanilla models, since the volatility term structure is a power function rather than a constant. [5] However, for calibration they employ techniques which might be too simplistic, like linear regression or which need global optimization and run the risk of getting stuck in local minima.

A different approach which is based on machine learning has recently gained prominence, especially with increase in availability of powerful computational resources. In particular, there have been works on using Artificial Neural Networks for Black Scholes option pricing and calibration. Notable among those are Horwaith et. al [3] and Itkin [4] . These have laid the theoretical foundations for proving important conditions of the neural network based pricers like no-arbitrage, stability etc, as well as compiling the tractable training approaches. But, the usage of machine learning techniques in fractal space for calibration, is little to none. This is a potentially rewarding approach.

The most attractive quality of such an approach is that most of the intensive computations are a one time activity. Adapting a Neural network based pricing model, and an inverse map, in the context of fractional derivative pricing [4] could make daily calibration of the model parameters, quicker and more efficient. While it needs to be explored and adapted to the fractal setting, the generalised version involves training an appropriately deep neural network based pricer with a sophisticated cost function having soft penalties to ensure no-arbitrage conditions are not (approximately) violated. This is done prior and stored. Later, an inverse map can be made use to obtain the calibrated parameters at once. In addition, invertible neural networks have had success in different domains including medicine and astronomy[7] , but haven't been used widely in the financial domain. It could also be successfully used for effective calibrations. However, more design analysis is required to confirm its plausibility.

In conclusion, academia has started giving importance to Black Scholes model's alternatives after realizing its numerous shortcomings. And the fractal approach is one of the most theoretically backed candidates. Tractable calibration techniques are imperative for any model's success. So, it would be fruitful to understand the existing estimation techniques in depth, and try to apply more advanced approaches such as Neural Networks to make the blooming field of Mandelbrotian derivative pricing practical and accurate.

1.4 Competitor Analysis

Option trading and modelling is an area which sees numerous continuous and fast improvements. While it is extremely difficult to compile all the important work done in this field, we have attempted to survey the existing work, in the light of the emergence of fractal market hypothesis and models based on it. The prime proposal is to examine the existing techniques and explore the use of advanced estimation techniques in the fractional Black Scholes model. The market viability of improving the calibration of this fractal model can be better grasped by making use of a SWOT analysis of the existing work. Here, an attempt is to elucidate the advantages and gains of the current work and approaches, along with what could be improved and what opportunities are present.

From the above analysis, few conclusions can be drawn. Firstly, the Black Scholes model has matured and has been a foundation for most of the option trading activities. But, many of its assumptions have been questioned and invalidated, especially after the black swan event of the 2008 global crisis. Fractal models, its viability and heuristic strengths have been discussed at some length. But, apart from basic estimation techniques for Hurst exponent and using the prevalent ideas of implied volatility, calibration of the Fractal models aren't given much attention, as clearly pointed out in recent works[9]. While there are clear benefits to doing this, it is to be noted that numerical or approximate techniques like Neural networks shouldn't undermine the heuristic and intuitive strength of the model by introducing instability or bias.

SWOT: Existing Research

Strengths	Weaknesses
• The option pricing based on Black-Scholes models are the current industry standard. • There have been various improvements to improve the shortcomings of the BS model. • Numerous analytical and numerical implementation strategies are available. • Calibration techniques are really mature and can be chosen according to specific needs. • Multiple theoretical advances, including strong proofs were made in the Fractal Market domain.	• Black Scholes model have been demonstrated to multiple theoretical limitations. • It has failed to predict Black Swan events. • Improvements of the model often come at a cost of less intuitiveness & computational complexity • While Fractional models are starting to be used, calibration techniques are still at an early stage.
Opportunities	**Threats**
• Fractional model based option pricing offers a heuristic approach, while being free of numerous erroneous assumptions. • Works have demonstrated superior performance to BS model and comparable performance to SABR model.[1] • If calibration techniques are improved for Mandelbrotian models, it could provide a practical yet more accurate forecasts. • Being based on heuristic techniques, it offers better feedback.	• Using machine learning models is more intense computationally. • Numerical techniques often require stability and bias analysis. • While more complex, SABR model offers slightly better fit and is more widely adopted.

Figure 1: SWOT analysis summary

2. Theoretical Framework

2.1 Inverse Map/ Invertible Neural Networks Approach

This approach is inspired by Deep Learning Inverse Map Approaches employed by Horvath. et. al (2019)[3] and Itkin (2019) [4] in the context of Vanilla Black Scholes model. Neural network based pricer plays an auxiliary role in calibration, but is often an effective primary step. So, that is discussed first.

2.1.1 Neural Network based Pricer

Analytical approaches are often time consuming and significantly slow down the calibration and estimation steps. For numerical advantages and tractability, many prefer approximation pricing models over the analytical models. The primary advantage of one of the more important approximation techniques, neural network based pricers is that the numerical approximation training of the pricing model becomes a one time preprocessing step which can be done offline. However, this is often the most computationally demanding step. Approximating the pricing functional using neural networks has numerous challenges like choosing a suitably rich model to learn its characteristics, curating big enough datasets, designing proper loss function to ensure essential market properties, and employing proper optimizers to avoid getting stuck at local minimas.

In the context of Fractal models, following in the line of Itkin at. al[4] , a straightforward approach could be to learn the characteristics by using random vectors generated by sampling the model. It should also be noted that preprocessing of such generated vectors are necessary for a quality dataset. One such step would be to avoid unnaturally large or small prices. While the network architecture is yet to be ascertained during training and finetuning, the number of layers should be at least 2 or more, to learn the analytical formula functional surface.

The 2 most critical pieces are that of cost function and training optimizer. All popular deep learning frameworks including Tensorflow and Pytorch provide efficient training optimizers. Cost function, on the other hand, is much more closer to the domain and hence, needs to be specifically designed. The primary component of the cost function is the mean squared error of the output price, but equally important is to ensure no arbitrage conditions are satisfied. The conditions for call prices as explained in Carr and Madan[8], are as follows:

$$\frac{\partial C}{\partial T} > 0, \quad \frac{\partial C}{\partial K} < 0, \quad \frac{\partial^2 C}{\partial K^2} > 0 \qquad \rightarrow (16)$$

But, as shown in Marquez-Neila et. al[7], in the field of Deep Learning, imposing hard constraints often adversely impacts the training effectiveness. Soft regularising constraints are more effective, when combined with modern backpropagation based optimizers. In short, the regularised loss function would be roughly of the form :

$$L_c = \arg\min_w \sum_i \sum_j \left\{ \left[C\left(\boldsymbol{\xi}_i\right) - C_{\text{ANN}}\left(\boldsymbol{\xi}_i, w_{i,j}\right) \right]^2 + \Phi_{\lambda_1,m_1}\left(-\frac{\partial^2 C_{\text{ANN}}\left(\boldsymbol{\xi}_i, w_{i,j}\right)}{\partial K^2} \right) \right.$$
$$\left. + \Phi_{\lambda_2,m_2}\left(-\frac{\partial C_{\text{ANN}}\left(\boldsymbol{\xi}_i, w_{i,j}\right)}{\partial T} \right) + \Phi_{\lambda_3,m_3}\left(\frac{\partial C_{\text{ANN}}\left(\boldsymbol{\xi}_i, w_{i,j}\right)}{\partial K} \right) \right\} \qquad \rightarrow (17)$$

where $\Phi_{\lambda,m}(x)$is a penalty function. A widely used penalty is :

$$\Phi_{\lambda,m}(x) = \begin{cases} 0, & x < 0 \\ \lambda x^m, & x \geq 0 \end{cases} \qquad \rightarrow (18)$$

2.1.2 Calibration techniques approaches

Calibration is an inverse problem of the above. Given the observable quantities and market prices, we wish to calculate the implicit parameters of the model. More often than not, calibration by inverting the analytical formula is infeasible. So, a data driven numerical approach is suited to this problem. The practitioners use model calibration with the recent data as often as possible to get best heuristics and parameter estimations. For this, price computation needs to be extremely efficient, which we can achieve using Neural networks based pricing, discussed above. The next and more important component is to find techniques for solving this optimization problem, expressed mathematically as:

$$\arg\min_{p} \sum_{i=1}^{n} \sum_{j=1}^{m_i} \omega_{i,j} \left\| C_M\left(\boldsymbol{\theta}_{i,j}\right) - C\left(\boldsymbol{\theta}_{i,j}, \boldsymbol{p}\right) \right\| \quad \rightarrow (19)$$

where θ refers to the observable quantities, p to the parameters, C_M to the market price and C to the model under consideration. That is, we aim to get constant values for the model parameters, H and σ, which is the best fit for the historical period under consideration.

Note that this optimization can be solved analytically under L^2 norm, with the parameters obtained using below formula:

$$p_k = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m_i} \bar{\omega}_{i,j,k} p_{k,\text{model}}\left(\boldsymbol{\xi}_{i,j}\right)}{\sum_{i=1}^{n} \sum_{j=1}^{m_i} \bar{\omega}_{i,j,k}} \quad \rightarrow (20)$$

The only portion left is to compute $p_{k,model}$. For obtaining $p_{k,model}$, we primarily attempted 2 related techniques. The first one is training of an inverse map which is a neural network on the lines of the pricing model analysed in the previous section. One attempts to train with just the generated data. This is often easier, but more challenging to train. The second approach is trained with an inverse dataset, formed by inputs and outputs of the pricer. The simplifying feature of these techniques that since the price incorporates the market requirements, this model can be trained in a relatively straightforward manner.

3. Methodology

Python programming language was used to do all components of implementation, including processing the data, implementing and training networks and to carry out algorithmic processes on our processed data. For neural network design, implementation and training, PyTorch deep learning framework was utilised.

3.1 Data Collection and Handling

Stock price data was obtained from Yahoo finance and option price data was obtained from Quantl. This data was the primary data source for modelling and testing the effectiveness of the modeling and calibration procedures.

In addition, python libraries like pandas, numpy etc were used for preprocessing of the market data and the generation of random vectors of observables, parameters and call prices sampling the fractal model,which were used for offline training of Neural network pricer.

3.2 Data Analysis and Modeling

3.2.1 Modeling Hurst exponent using Rescaled Range (R/S) analysis

The techniques explained in the literature review on modeling Hurst exponent using R/S method were used. We used a linear model fitting method to determine the hurst exponent of our stock price time series data.

The first approach was to compute the Hurst exponent using (R/S) analysis for the whole data set from the start to the end of the time series data.

We also in a second approach implemented an algorithm to estimate the hurst exponent for each 30 day period and we generated a monthly hurst exponent time series data set which we then used to predict the hurst exponent for the month in which the call options in the price chain data were priced. The procedure deliberately left out the stock price data for the month we intended to predict such that the value we obtain would be the estimated Hurst exponent value for the target month.

3.2.2 Computing monthly (30 day period) Hurst exponent using (R/S) analysis and predicting next period value

One method was to get an average for all of the monthly hurst exponent values we computed using R/S analysis, the second approach was to fit an Autoregressive (AR) model to monthly hurst exponent data as we observed that even though there were month to month differences in the value of the hurst exponent computations, the data appeared to oscillate about a constant mean. The AR model was then used to predict the hurst exponent value for our target month.

3.2.3 Computing implied volatility of our stock data

Implied volatility was computed from the hurst exponent and stock returns of the series, note that we used two day log returns instead of daily log returns simply because in computation of volatility as per equation (14) a value of one (1) for the variable (dt) results in the same volatility value for all the various models even if we generate different values of the hurst exponent as the stock return is the same for each given stock. This is because one(1) raised to any power is always one. Two day stock return was able to give us accurate volatility estimates.

3.2.4 Comparison of model effectiveness using Mean Square Error Analysis

We used closed form solutions in equation (4) and (5) to price our options using the Fractional BS model formulas, option price variables like interest rate, strike price and duration of the options were obtained from the options price chain for 15th of november 2019. We also computed the option prices using the standard BS model.

The computed prices are then compared to the actual option prices and we use mean square error(MSE) to determine the absolute error in our pricing as compared to actual option prices.

3.2.5 Comparison of Classical solution and Neural network pricer

The option prices obtained from the two methods were compared with actual market data option prices to determine the effectiveness of each of the models and to notice any areas of improvement that can be made on each of the models.

3.3 Calibration using Neural Networks

Generated random stock parameters were used for offline calibration of our neural networks. The methods used are as per explanation in the theoretical framework. For implementation, the primary tool was PyTorch for NN implementations. There were preliminary attempts using FrEIA(Framework for Easily Invertible Architectures) for designing Invertible Neural Networks. But this was discontinued due to implementation difficulties. Multiple approaches of training an inverse pricing map were considered, while trying to utilise the effectiveness and understanding its pitfalls.

3.3.1 Option Inverse Pricer Architecture design

While designing the neural network, the items we considered were the architecture, the loss function and training optimiser.
A depth of 3 was chosen to utilise the approximation power of deep Neural Networks. The architecture is as follows:

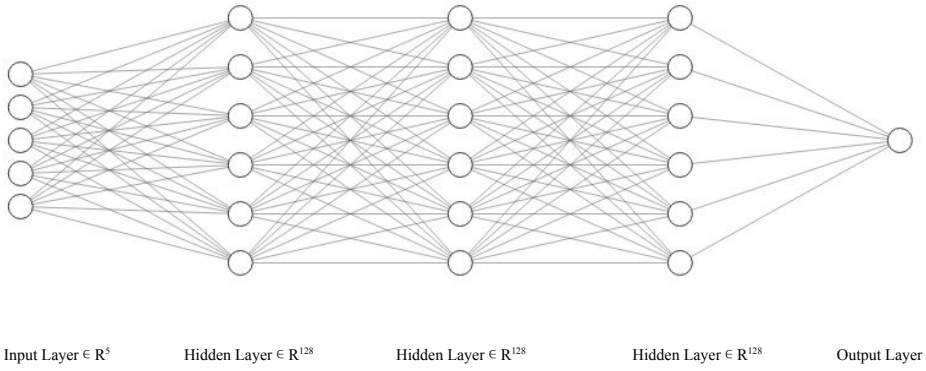

Figure 2: Pricer Neural network layer architecture

The activation function which is essentially for the neural networks to approximate a nonlinear function needs to be carefully chosen since the additional no-arbitrage conditions which can be imposed requires the functional to be C^2 i.e it should be twice continuously differentiable. So, activation functions like RELU (Rectilinear Linear Unit) and ELU (Exponential Linear Unit) , while proven to be effective in vanilla networks, don't qualify. Ikin suggests using a modified ELU (MELU) which was tested, but the training was unstable. A relatively new C^2 activation function called Gaussian Error Linear Units (GELU) was used in conjunction with the softplus function which resulted in stabilised training[16]. It was chosen specifically to allow for applying different orders of derivatives. The various activation functions considered are plotted for comparison below. The loss function would be Mean Squared Loss.

An important problem, which hasn't been actively considered in the fractional Brown Scholes model based pricing, is the calibration of its parameters Hurst parameter (H) and fractional volatility (σ). Often, the computations are a time consuming step in this. Neural networks can significantly reduce the time involved due to quick inference.

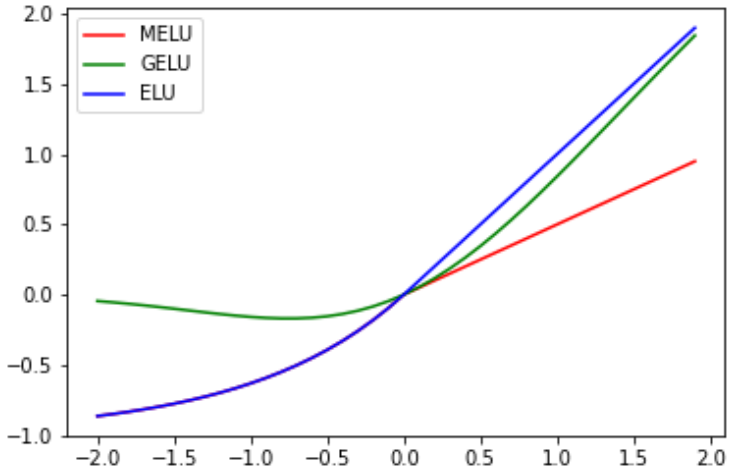

Figure 3: Activation function comparison plots

3.2 Market data Calibrations

The Neural network framework developed in the prior sections was used for market data based calibrations. For calibration, option prices as well as the market observables for the historically relevant period are used to find the most suitable model parameters. The best H and σ (constants) estimates were obtained using the two techniques outlined below to obtain the model parameters using market data.

3.2.1 Finding H and σ individually

1. We use the inverse mapping neural network to infer the value $\sigma\tau^H$ for all available market data observables and option prices (C,S,K,r,τ). This is extremely quick as it is just a forward pass of the neural network.
2. we then fetched the values $\sigma\tau^H$ computed in the previous step and τ we then. apply log transformation on the obtained data. This gives the set (log(τ), log(σ) + H*log(τ)).
3. We then used linear regression to obtain the best fitting H and σ.
4. For forecasting,we used a neural network pricer which accepts (S,K,r,τ,σ,H) and returns the call price. Note that in this step, market observables like S,K,r,τ were obtained from the option price chain data we obtained from Quantl and the parameters σ,H were then obtained as per step 3.

3.2.2 Finding the best fitting $\sigma\tau^H$

1. We used the inverse mapping neural network to infer the value $G = \sigma\tau^H$ for all available market data observables and option prices (C,S,K,r,τ).
2. The calibrated parameter in this case was computed analytically as it is an optimization of the below form :

$$\arg\min_{p} \sum_{i=1}^{n} \sum_{j=1}^{m_i} \sum_{k=1}^{N} \bar{\omega}_{i,j,k} \left\| G_{k,\mathrm{ANN}}\left(\boldsymbol{\xi}_{i,j}\right) - G_k \right\| \qquad \rightarrow (21)$$

The parameters results are shown below, and a similar setup has been worked out in detail in Itkin was adopted.

$$G_k = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m_i}\bar{\omega}_{i,j,k}G_{k,\mathrm{ANN}}\left(\boldsymbol{\xi}_{i,j}\right)}{\sum_{i=1}^{n}\sum_{j=1}^{m_i}\bar{\omega}_{i,j,k}} \quad \rightarrow(22)$$

$$\text{where } \bar{\omega}_{i,j,k} = \left|\frac{\partial C_{\mathrm{ANN}}\left(\boldsymbol{\theta}_{i,j},\boldsymbol{p}_{i,j}\right)}{\partial p_{k,\mathrm{ANN}}\left(\boldsymbol{\xi}_{i,j}\right)}\right| \quad \rightarrow(23)$$

These derivatives are already computed by the neural network during backpropagation passes, and are available directly for use under torch.grad in Pytorch.

4. Results

4.1 Hurst Exponent techniques using Rescale analysis method and next period Predictive techniques.

In this section, we do comparison of 4 different models which we used for determining hurst exponent and the implied/fractional volatility, in our analysis we used 3 different stocks namely Agile technologies(A), Amazon(AMZN) and ACCO brands corporation(ACC)). The choice of these companies is so as to avoid picking stocks that are from the same sector and thus too correlated and therefore might present similar results and the price propagation in the same sectors tends to be similar

In our approach we used the commonly used Rescale range (R/S) Analysis method we then also introduced prediction of next period(month) hurst exponent value using process averaging and also fitting an Autoregressive (AR) model to the computed monthly Hurst exponent from the stock price time series data.

The approaches and results are discussed below:

1) Using Rescaled range Analysis for all the data period for entire time series data period

We computed computed $log((\frac{R}{S})_n)$ against $log(n)$ and used train_test split to create both training data and test data. The training data is fit into a linear regression model in python Scikit-learn and a scatter plot and line of best fit is obtained as below:

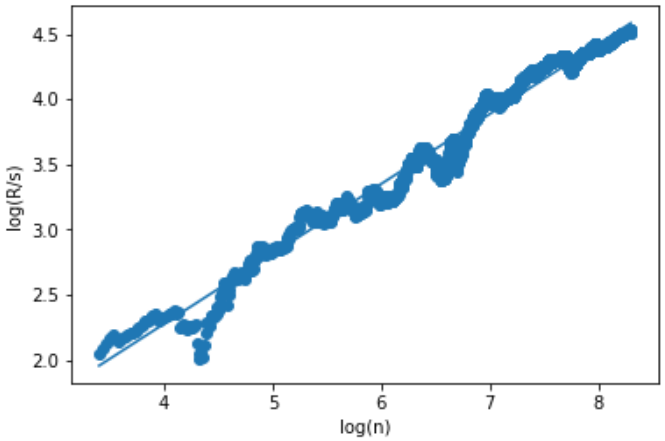

Figure 4: Linear model to fit Agile Technologies(R/S) log plot data

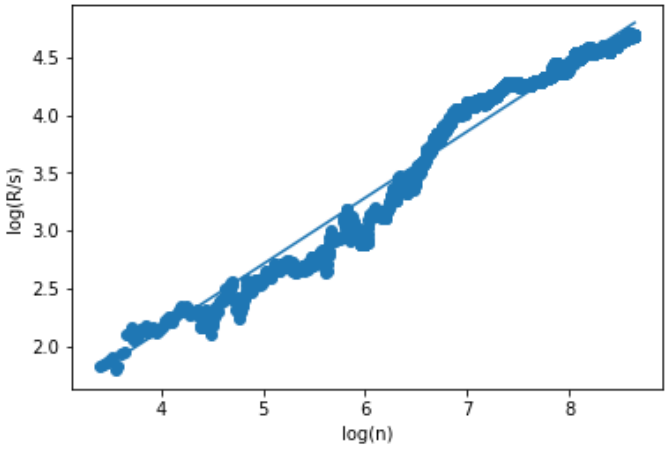

Figure 5: Linear model to fit Agile Technologies(R/S) log plot data

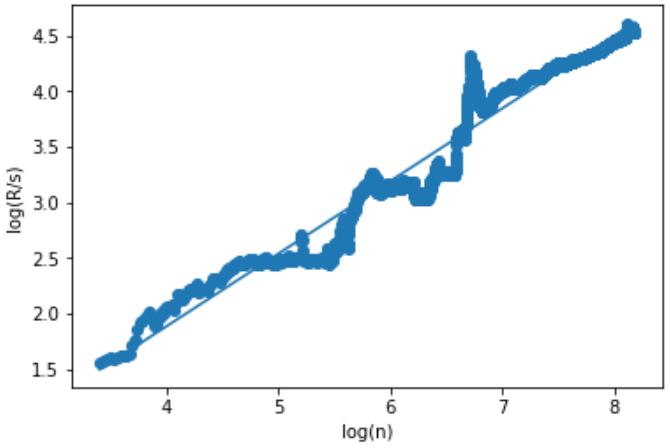

Figure 6: Linear model to fit ACCO (R/S) log plot data

2) Compute Hurst exponent for each of the months from start to a month before option pricing month as per the pricing data in the options chain we obtained from Quantl . We computed hurst exponent values for each month using rescaled analysis, the data for the 3 different stocks is plotted as below.

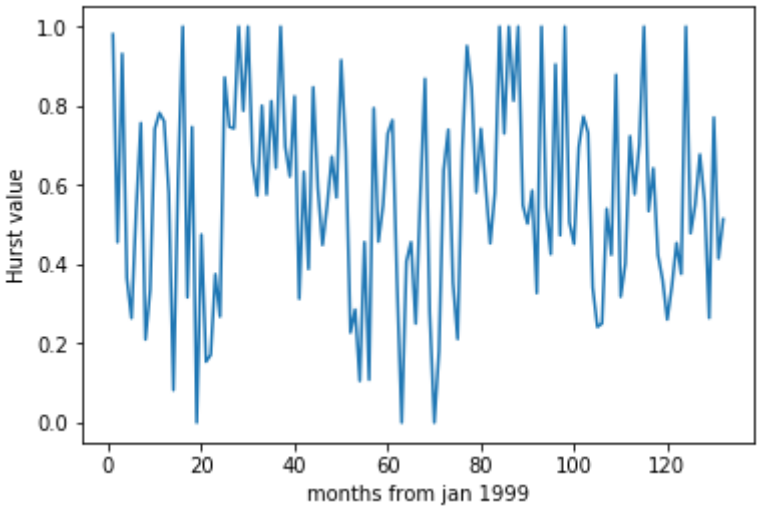

Figure 7: Monthly hurst exponent data for Agile Technologies

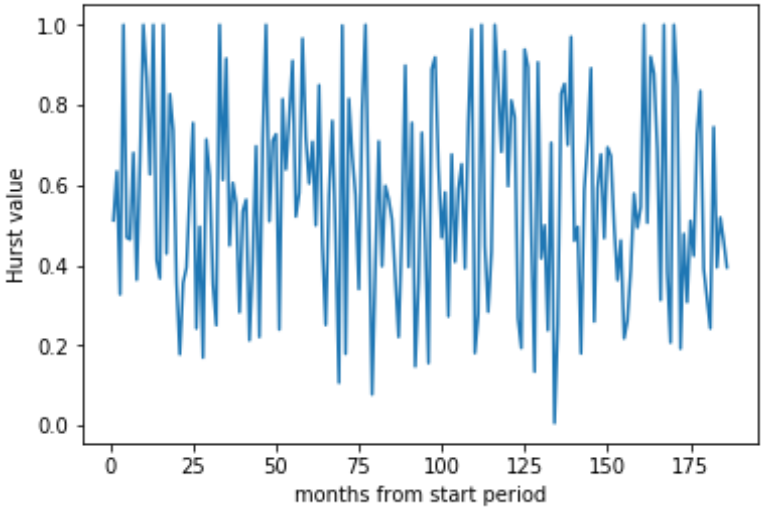

Figure 8: Monthly hurst exponent data for Amazon

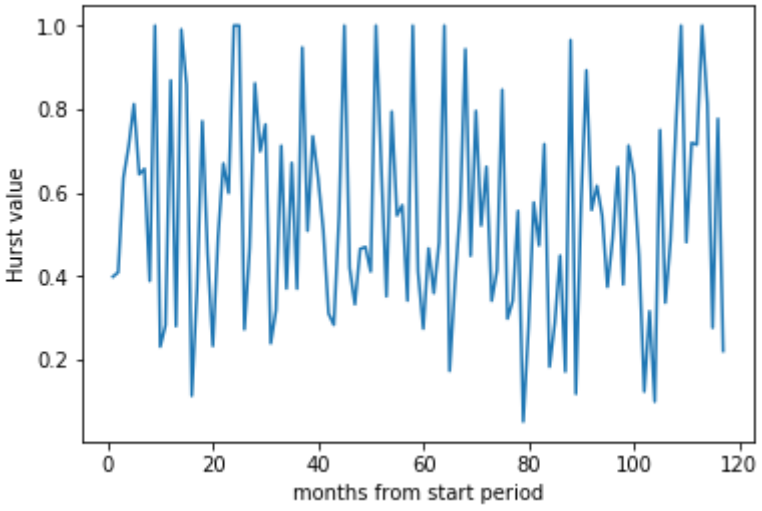

Figure 9: Monthly hurst exponent data for ACCO

One approach was computing the average for all the computed monthly Hurst exponent data; this average is the prediction of the Hurst exponent value for the option pricing month. The predicted Hurst exponent is then used to price the options using closed form formula as per equations (4) and (5).

Looking at the graph plots we notice that the time series plots of the computed monthly data exhibits characteristics of stationarity as it appears to have a constant value through the months with periods of volatility, we therefore used an autoregressive model to model the evolution of the hurst exponent over the months model and use this model to predict the next period hurst exponent which is then used to price our options.

Below is the summary of the models used to fit our monthly hurst exponent data :

Out[22]:

ARMA Model Results

Dep. Variable:	y	**No. Observations:**	132
Model:	ARMA(2, 0)	**Log Likelihood**	-3.112
Method:	css-mle	**S.D. of innovations**	0.248
Date:	Mon, 10 Aug 2020	**AIC**	14.223
Time:	00:39:26	**BIC**	25.755
Sample:	0	**HQIC**	18.909

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.5698	0.029	19.412	0.000	0.512	0.627
ar.L1.y	0.1585	0.087	1.820	0.071	-0.012	0.329
ar.L2.y	0.1098	0.087	1.262	0.209	-0.061	0.280

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	2.3809	+0.0000j	2.3809	0.0000
AR.2	-3.8246	+0.0000j	3.8246	0.5000

Figure 10: AR model to fit generated Agile Technologies monthly hurst exponent data

Out[47]:

ARMA Model Results

Dep. Variable:	y	**No. Observations:**	186
Model:	ARMA(2, 0)	**Log Likelihood**	-3.325
Method:	css-mle	**S.D. of innovations**	0.246
Date:	Mon, 10 Aug 2020	**AIC**	14.650
Time:	01:22:33	**BIC**	27.553
Sample:	0	**HQIC**	19.879

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.5654	0.015	36.676	0.000	0.535	0.596
ar.L1.y	-0.0380	0.073	-0.523	0.601	-0.180	0.104
ar.L2.y	-0.1353	0.072	-1.869	0.063	-0.277	0.007

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	-0.1404	-2.7151j	2.7187	-0.2582
AR.2	-0.1404	+2.7151j	2.7187	0.2582

Figure 11: AR model to fit generated Amazon monthly hurst exponent time series data

Out[74]:

ARMA Model Results

Dep. Variable:	y	**No. Observations:**	117
Model:	ARMA(2, 0)	**Log Likelihood**	-2.324
Method:	css-mle	**S.D. of innovations**	0.247
Date:	Mon, 10 Aug 2020	**AIC**	12.648
Time:	01:27:15	**BIC**	23.696
Sample:	0	**HQIC**	17.133

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.5452	0.020	27.574	0.000	0.506	0.584
ar.L1.y	-0.0914	0.093	-0.986	0.326	-0.273	0.090
ar.L2.y	-0.0648	0.093	-0.699	0.486	-0.246	0.117

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	-0.7050	-3.8643j	3.9281	-0.2787
AR.2	-0.7050	+3.8643j	3.9281	0.2787

Figure 12: AR model to fit the generated ACCO monthly hurst exponent time series data

All the 3 models are well fitting as the explanatory variables, which are greater than twice the standard error.

The models are encompassing the data and we then use them to predict pricing month hurst exponent values.

4.2 Using Our models to test the accuracy in pricing of the fractal BS model to the standard BS model

The computed option prices we obtain using our predicted and computed hurst and volatility values are used to price our options and then are compared to the actual option price data from the options chain for each stock.

Below is a summary of the performance of each of the models:

Company: Amazon **Stock ticker: AMZN**

Model	Hurst exponent	Volatility	MSE of prices
Linear model for R/S	0.574180506	0.035246021	192.187123
Monthly hurst computation-avg	0.565174038	0.035466744	192.528175
Monthly hurst with AR model	0.5858451625	0.034962195	191.858595
Standard BS model	N/A	0.0370501	196.781412

Company: Agilent Technologies Inc. **Stock ticker: A**

Model	Hurst exponent	Volatility	MSE of prices
Linear model for R/S	0.535963462	0.02874081	2.435303
Monthly hurst computation-avg	0.569142123	0.02808738	2.448801
Monthly hurst with AR model	0.543859960	0.02858393	2.437842
Standard BS model	N/A	0.04257024	2.459883

Company: ACCO brands corporation Stock ticker: ACCO

Model	Hurst exponent	Volatility	MSE of prices
Linear model for R/S	0.65193986	0.035514431	0.012890
Monthly hurst computation-avg	0.54466997	0.038255714	0.014628
Monthly hurst with AR model	0.55985568	0.037855149	0.014393
Standard BS model	N/A	0.039742610	0.015175

We notice that for each of the modeling approaches including the predictive approaches of obtaining the time series Hurst exponent and volatility data, option pricing using the fractal Black Scholes model performed better than the standard BS model when compared to actual option price market data.

4.3 Neural Networks Approach to Hurst exponent estimation

The first approach was to construct and train the neural network which is essentially a map of market observables and the option price to return the model parameters, i.e. a function f:(C,S,K,r,τ) -> (H,σ). But, these gave close to random predictions. This could be explained when the underlying model is analysed. H and σ are independent parameters which essentially makes the possible values a functional expression rather than constants for a given set of inputs. So, the neural networks predictions are in a faulty setup.

4.3.1 Applying Fractional Model Insights

A domain specific simplification was done to achieve tractable results. The model and the simulated data was modified such that instead of predicting H and sigma, the expected output would be the functional $G:\sigma\tau^H$ which is constant for a given set of C,S,K,r,τ. This is a sufficient output since the call prices in eq (4) can be rewritten as a function C = f(S,K,r,τ,G) owing to the fact that H and σ only appear as expressions of G.

First attempt under this framework was to model f:(C,S,K,r,τ) -> G directly. This gave a decent fit, but the training was not very stable. Continuous monitoring of each epoch and reducing learning rate, similar to gradient clipping. Below is the in-sample and out of sample predictions. There is sufficient generalization but a bit of room to improve. Using Neural networks to directly return the calibrated parameters for Black Scholes model has been cited by few authors as difficult to train (Tomas[3]). It is reflected even in the fractional Black Scholes model.

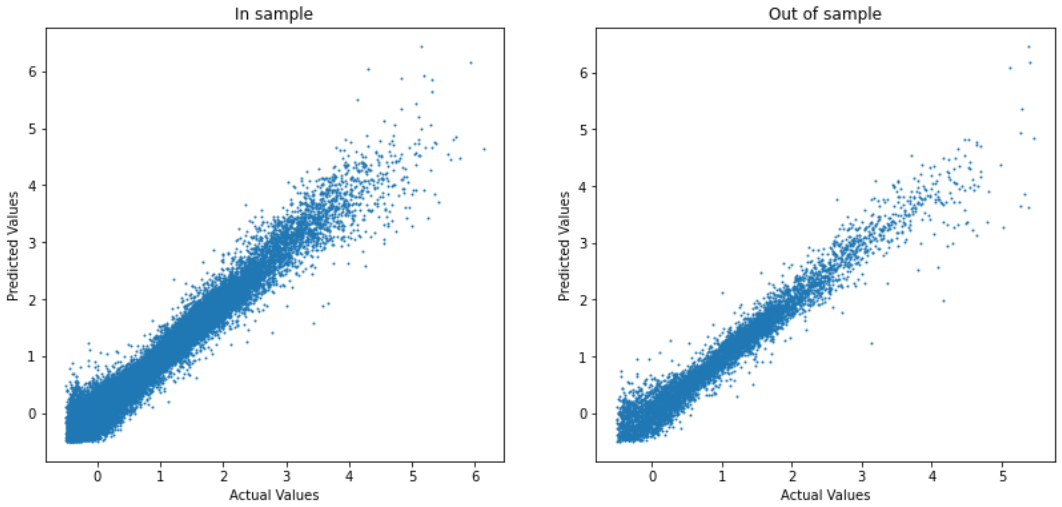

Figure 13: Inverse Mapping model data fit comparison

4.3.2 Second-Leg Approach of using a Pricer and Inverse Mapping

A second approach would be to train an intermediate call price approximator and use its predictions instead of the simulated prices to train the inverse mapping network. Using the pricing maps outputs as opposed to the actual prices has been shown to be more robust (Horvath), wherein the first steps involves approximating the neural network pricer and secondly using its outputs to solve the optimization problem of finding the calibrated parameters.

Training this neural network based pricer can be done using the above network since it is structurally similar and of the same complexity. Initial experiments could be simplified as follows: Since the call pricer is linear homogenous in the stock price S and strike price K, K could be eliminated by scaling the dataset: $S \rightarrow S/K$, $C \rightarrow C/K - \min(C/K) - 0.5$ along the lines of Culkin et. al's approach[17]. Essentially, this ANN is a function f: $(S,T, r, H,\sigma) \mapsto C_f$. The initial experiments used just this vanilla loss function with no penalty and had a good fit as shown below.

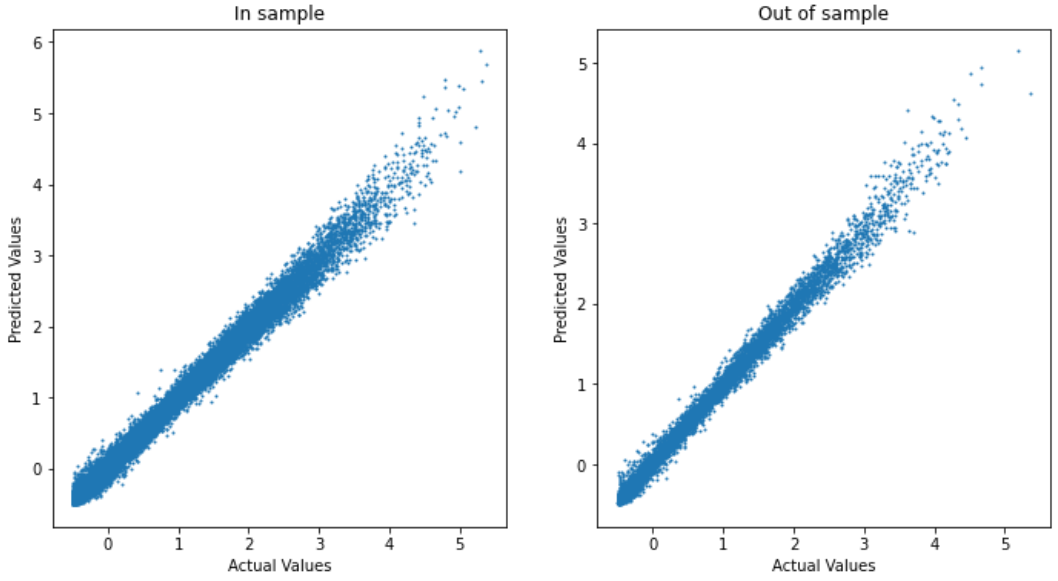

Figure 14: Vanilla pricing model fit comparison

But, as Carr and Madan (2005) proves, any call pricing model should satisfy the three conditions mentioned in Equation (12). In almost all deep learning models, additional constraints like these will be better achieved by using soft constraints like penalisation as opposed to hard constraints. It is demonstrated in Marquez-Neila et al (2017). But, it is to be noted that this does not eliminate arbitrage, rather made negligible. However to enforce these constraints involving derivatives and double derivatives w.r.to K, we need to increase the input dimension back to 6. This results in a slightly inferior fit after an initial round of training as shown below. But as expected, the penalty component of L_c in equation (13) is very less compared to the original model. And by adjusting the λ to 0.05, there is a good trade-off between the fit and penalty constraints as shown in the diagram.

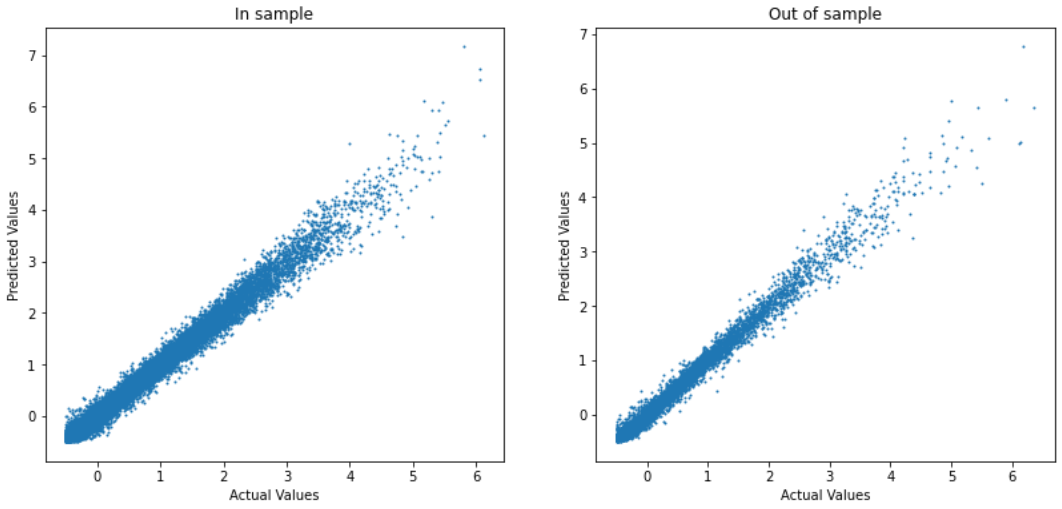

Figure 15: No arbitrage constrained pricing model fit comparison

The second leg of this technique is to capture its results and replace the simulated call price values in the dataset with this value. Now, this modified dataset can be used to train the former inverse pricing network. As expected, the training is more robust and reaches the optimum quickly. But the results are more grainy. This is possibly due to noise from the pricer results, which are compounded during the second network training. There is some room for improvement here.

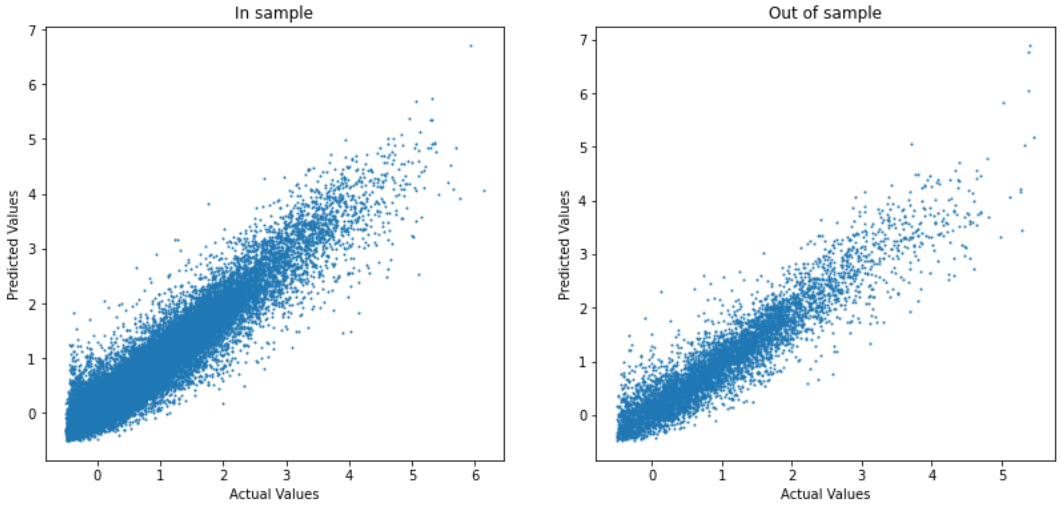

Figure 16: Second leg inverse map sample data model fit comparison

4.3.3 Validation of Neural Network Calibration Proposals

Although the neural networks have good performance in the offline steps, it is clear that a bit more tighter fit is required for actual market usage. This could be due to chained approximations resulting in noise additions. To test this, a test set of 1000 was generated to apply the calibration proposal mentioned in Section 3.2.1. The market data for testing the effectiveness of the calibration needs to be standardised for better comparison with other standard techniques. However, a significant problem we encountered was getting enough historical options data.

Following the steps which involve inference of $G = \sigma\tau^H$, log transformation of G and linear regression based fitting, comparable results are obtained. But there is significant noise which needs to be eliminated.

	Generated data - hidden parameters	Parameters obtained after direct calibration	Parameters obtained after 2-step calibration
H	0.5	0.42	0.415
σ	0.1	0.12	0.138

4.3.4 Further improvements

The inverse map neural network can certainly be improved, possibly by using better data generation and a more expressive neural network architecture. The historical data can be leveraged using neural network techniques like Long Short term memory (LSTM) , Gated Recurrent Units (GRU) etc.

A more challenging, but intuitive approach would be to use a different architecture called Invertible Neural Networks, introduced by Ardizzone et. al (2018)[9] for pricer. Although the model complexity increases, it gains invertibility property. And they could be developed using the FrEIA framework built on PyTorch. But, they are less understood, difficult to train and have less framework support especially for penalised loss functions. Our preliminary attempts did not yield any noteworthy results.
This approach and its proposals should also be validated using long term market data to verify its effectiveness in estimation of the model parameters.
Another area of development is the use of the models in the prediction of price movements compared to predictions done by the standard BS model. This is also left for future research.

Lastly, an interesting combination approach would be to check how such models can be integrated with Fast Fourier transforms (FFT) to even increase the speed of computations of the option prices although much trade off of accuracy for speed should be minimised as much as possible.

5. Conclusion

We verify that the Models based on fractal analysis reduce the error between the actual price and the calculated price compared to the standard BS model. Even the next period predictions were able to perform better than the standard BS model in the pricing of options when compared to actual market data. This approach can therefore be explored to generate profitable trading strategies(alpha generation).

And there has been significant progress in the implementing and training of option pricing neural network models, as well as the implementation of pricing maps. The offline training steps have been documented in the codebase and can be easily reused for variations of other fractal based models. The inferences as well as the analytical computation of calibration are extremely quick, which is an important advantage in finance. These Neural network models can find use in high frequency trading (HFT) applications where speed is of the essence. Finally, the proposals for fractional Black Scholes model calibration using neural networks are shown to be theoretically backed, fast and scalable.

However few important steps still remain. The performance of the inverse mapping needs to be improved further to get market ready calibration, which could be achieved both by better architectures, and specialised training. The neural network calibrations were posited with simulated data, but market data analysis needs to be done to affirm the conclusions.

6. References

Various Journals, Books and Softwares used for the fulfillment of this project.

6.1. Journal Articles

[1] Oleksii Romanko (2018). Option pricing with fractals: An empirical analysis. Economic Analysis MA Thesis. Kyiv School of Economics

[2] Dupire, B (2006). Model free results on volatility derivatives. Working paper. Bloomberg, NY: SAMSI.

[3] Horvath, Muguruza, Tomas (2019). Deep Learning Volatility A deep neural network perspective on pricing and calibration in (rough) volatility models. arXiv:1901.09647

[4] A. Itkin (2019). Deep learning calibration of option pricing models: some pitfalls and solutions. arXiv:1906.03507.

[5] Flint, Emlyn & Mare, Eben (2017). Fractional Black–Scholes option pricing, volatility calibration and implied Hurst exponents in South African context. South African Journal of Economic and management Sciences. 20. 10.4102/sajems.v20i1.1532.

[6] Li, Kinrey & Chen (2014). Implied Hurst Exponent and Fractional Implied Volatility: A Variance Term Structure Model. Available at SSRN: https://ssrn.com/abstract=2383618

[7] Marquez-Neila, P., Salzmann, M., Fua, P., 2017. Imposing hard constraints on deep networks: Promises and limitations. URL: https://arxiv.org/pdf/1706.02025.pdf

[8] Carr, P., Madan, D., 2005. A note on sufficient conditions for no arbitrage. Finance Research Letters 2

[9] L. Ardizzone, J. Kruse, S. Wirkert, D. Rahner, E. W. Pellegrini, R. S. Klessen, L. Maier-Hein, C. Rother and U. Kothe (2018), Analyzing inverse problems with invertible neural networks. arXiv:1808.04730

[10] Angkola, F. (2016). A generalized fractal dynamics option pricing model with transaction costs. PhD thesis. Curtin University

[11] Teneng, Dean. (2011). Limitations of the Black-Scholes model. International Research Journal of Finance and Economics. 99-102.

[12] Hu, Y. and B. Oksendal, (2000), Fractional white noise calculus and application to Finance, Preprint, University of Oslo

[13] Hu, Y., B. Oksendal, and A. Sulem, (2000), Optimal consumption and portfolio in a Black-Scholes market driven by fractional Brownian motion, Preprint 23/2000, University of Oslo

[14] Lianga, Z., Fengb, Z., and Guangxianga, X., (2012),Comparison of Fractal Dimension Calculation Methods for Channel Bed Profiles, International Conference on Modern Hydraulic Engineering

[15] Feng, Z., Pettersson, R., (2018). Stock-Price Modeling by the Geometric Fractional Brownian Motion A View towards the Chinese Financial Market.Degree project. Linnaeus university Sweden.

[16] D. Hendrycks , K. Gimpel, (2020), Gaussian Error Linear Units (GELU). URL: https://arxiv.org/pdf/1606.08415.pdf

[17] R.Culkin, R.Das (2017), Machine Learning in Finance: The Case of Deep Learning for Option Pricing, URL: https://srdas.github.io/Papers/BlackScholesNN.pdf

6.2. Softwares

[18] Codecogs plugin for rendering LaTeX equations. Found here: https://www.codecogs.com

6.3. Books

[19] Marcos Lopez de Prado, Author, “Advances in Financial Machine Learning”, Wiley Publishing. (2018).