PyTorch project

Pricing options and computing implied volatilities using neural networks

Done by : ABBADI Khaoula

MASEF 2022/2023

Table of content

1 Introduction and context

Neural networks have a lot of applications, specifically in finance; from pricing, hedging, or even numerically approximating PDEs. The general goal of this project is to use an ANN to answer to one of the different questions in finance.

When it comes to ANN option pricing, we can either fit an ANN-based regression model to market data, i.e. option prices and underlying values, or we can improve the performance of *model-based* pricing with ANNs, i.e. accelerating the classical PDE solvers by replacing them with the ANNs. This project is the latter.

In this context, I chose to work on the use of ANNs in pricing European options under two different models: the Black-Scholes and the Heston asset price models. Furthermore, we will also build an ANNs to recreate the volatility surface in a data driven approach. Our guide for this project is the paper titled *Pricing options and computing implied volatilities using neural networks* by Shuaiqiang Liu, Cornelis W. Oosterlee and Sander M. Bohte.

There exist numerous models for asset prices in financial markets, ranging from ones that are easily implemented to ones which pose more difficulties. The models we'll be focusing on are representative of these two tendencies. In section 2 we will detail the two model prices used in this project, in section 3 we will look at the implementation of these models to obtain training data for our ANNs in addition to the general architecture and hyperparameter choices for the ANNs, in section 4 we will present the different results obtained for each ANN class and lastly in section 5 we will examine our findings and propose different ways this project could be made better.

2 Asset price models

A European option is defined by two key elements: the strike price K and the maturity T. Under the risk neutral measure, the price of a European option is simply given by the expectation under this measure of the payoff at T conditioned on the filtration at that time, i.e. in the case of a call option for example, the price at time t is given by $e^{-r(T-t)}\mathbb{E}^{\mathbb{Q}}[(S_T - K)_+ \mid \mathcal{F}_t]$ where $\mathbb{Q}$ is the risk neutral measure (also known as the equivalent martingale measure) and $\mathbb{F} = (\mathcal{F}_t)_t$ is the filtration on the usual probability space. We introduce the time to maturity $\tau = T - t$ which will be used in formulae to make them less heavy.

One way to compute this conditional expectation is to make an assumption about the model of asset price S.

2.1 The Black-Sholes asset price model

The Black-Scholes asset model, also called the geometric Brownian motion (GBM) asset model, models the underlying asset price as a log-normal variable. This choice of modeling allows us to obtain explicit formulas for pricing, at least in the case of European options.

The SDE followed by the asset price, which is denoted as $(S_t)_{t\geq 0}$ is the following:

$$dS_t = \mu S_t dt + \sqrt{v} S_t dW_t^S$$

μ : is the drift of the asset price.

$\sqrt{\nu}$: is the volatility of the asset price, also noted as σ where ν is the variance.

$(W_t^S)_{t\geq 0}$: is the standard Brownian motion associated with the asset price.

Given that there exists a strong solution to this SDE, we can use the Feynman-Kac formula to find the associated pricing PDE. Denoting the option price as $V(t,S)$ we can write:

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

Where r is the risk-free interest rate, which corresponds to the drift of S under the risk-free probability measure.

To solve the PDE, we need to specify a final solution corresponding to a specific payoff, for example in the case of a European call option we have $V(t = T, S) = (S_T - K)_+$which is the payoff of the European call option at maturity.

Given this condition, the PDE can be solved analytically, and the solution is given by:

$$V_c(t,S) = S\mathcal{N}(d_1) - Ke^{-r\tau}\mathcal{N}(d_2)$$

Where $d_1 = \frac{\log(\frac{S}{K}) + (r - \frac{\sigma^2}{2})\tau}{\sigma\sqrt{\tau}}$, $d_2 = d_1 - \sigma\sqrt{\tau}$ and $\mathcal{N}(.)$ is the CDF of the normal distribution.

In our data generation, the prices for various values of the parameters will be computed using this formula.

A major advantage of the GBM asset price model is the existence of this closed formula. However, one of the key assumptions of this model is that the volatility of the asset price movement is constant. This is proven to be false namely because of the observed volatility skew; indeed, the volatility is **not** constant across strikes and maturities. The next model we'll be looking at allows us to recreate these stylized statistical properties of the volatility.

2.2 The Heston asset price model

The Heston model is part of a larger class of models called the *Stochastic Volatility (SV) models.* The SV models allow us to step away from the assumption of a constant volatility by modeling as a diffusion process. In the specific case of the Heston model, the variance process $(\nu_t)_t$ is driven by a Brownian motion that is correlated with W^S, this gives a system of SDEs, and we write under the risk neutral measure:

$$\begin{cases} dS_t = \mu S_t dt + \sqrt{\nu_t} S_t dW_t^S, \qquad S_{t_0} = S_0 \\ d\nu_t = \kappa(\bar{\nu} - \nu_t)dt + \gamma\sqrt{\nu_t} dW_t^\nu, \nu_{t_0} = \nu_0 \\ dW_t^S dW_t^\nu = \rho dt \end{cases}$$

ρ : is the correlation coefficient of two Weiner processes.

κ : is the mean reversion speed of the variance process.

$\bar{\nu}$: is the long-term variance.

γ : is the volatility of the volatility (or the volatility of the variance).

Similarly, we can derive the multi-dimensional pricing PDE:

$$\frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \kappa(\bar{\nu} - \nu)\frac{\partial V}{\partial \nu} + \frac{1}{2}\nu S^2\frac{\partial^2 V}{\partial S^2} + \rho\gamma S\nu\frac{\partial^2 V}{\partial S\partial \nu} + \frac{1}{2}\gamma^2\nu\frac{\partial^2 V}{\partial \nu^2} - rV = 0$$

This model obviously has more parameters than the GBM model, but it is precisely by varying the added parameters $\kappa, \rho, \gamma, \nu_0, \bar{\nu}$ that we can successfully recreate a wide variety of volatility surfaces. On the other hand, the Heston PDE does not have an analytical solution, therefore the solution must be approximated numerically.

In accordance with the authors, we use the Fourier Cosine Expansion method, also known as COS method [1], to compute the prices under the Heston Model.

2.3 Implied volatility

The implied volatility for a given strike and maturity (or equivalently for a given moneyness and time to maturity), denoted by σ^*, is the value of the volatility σ that equalizes the theoretical option price and the one observed on the market. In other words, if we denote the theoretical option price under the GBM model as $BS(\sigma; S, K, \tau, r)$, and $V^{market}(S, K, \tau)$ as the corresponding observed market option price, then σ^* is the unique solution to the root finding problem:

$$BS(\sigma^*; S, K, \tau, r) - V^{market} = 0$$

Thanks to the monotonicity of the Black-Scholes formula with respect to the volatility parameter, we know that a solution exists and is unique. Although the problem can't be solved analytically, it can be solved using a numerical iterative technique. In our case, we will use the method `brentq` from `scipy.optimize` in keeping with the authors.

3 Data generation and the ANNs

To generate training data for the ANNs, we need to sample from the feature space in each model and then price accordingly. We kept the feature space as defined by the authors and used the Latin Hypercube Sampling method to sample 1 million data points from the feature space of the Black-Scholes and Heston ANNs. The models are subsequently trained on 90% of the dataset, and the remaining is used for testing.

When it comes to the implied volatility ANN, there are multiple ways to go about learning the relationship between the price and the implied volatility, in our case we used the prices we computed for the Heston ANN as the "market price" in the root finding problem, and then we used Brent's algorithm to get the implied volatilities for each price (so 1 million data points in this case as well). Additionally, in the last ANN created, we used a modified price as input for the model to try to improve the model performance when it comes to recreating the volatility surface, namely we use the price given by:

$$\tilde{V} = V_t - \max\left(S_t - Ke^{-r\tau}, 0\right)$$

As for the ANN hyperparameters, the authors conducted a hyperparameter optimization procedure to find the optimal ones. We used the optimal values they found when it was feasible, and we chose more convenient values when not (specifically in the case of the number of epochs).

The following table summarizes all the ANNs that can be found in the notebook in the order they are in:

	BS	**Heston**	**Implied Volatility**		
Feature space intervals	Moneyness : [0.4, 0.6] Time to maturity : [0.2, 1.1] Risk-free rate : [0.02, 0.1] Volatility : [0.01, 1.0]	Moneyness : [0.6, 1.4] Time to maturity : [0.1, 1.4] Risk-free rate : [0.0, 0.1] Correlation : [-0.95, 0.0] Mean reversion speed : [0.0, 2.0] Long term variance : [0.0, 0.5] Volatility of the variance : [0.0, 0.5] Initial volatility : [0.05, 0.5]	The price given by the Heston model	The price given by the Heston model + all the features of the Heston ANN	The "flattened" price given by the Heston model + all the features of the Heston ANN
Output	Scaled price V/K	Price V	The BS implied volatility		
Hidden layers	4				
Activation function	ReLU				
Neurons	400				
Epochs	50	100	60		
learning rate	Decaying schedule where the rate is divided by 10 every 10 epochs starting from 10e-4				
Parameter initialization	Glorot uniform				
Batch size	1000				
Loss function	MSE				
Optimizer	ADAM				

Table 1. Summary of all features and hyperparameters of the ANNs

4 Results

4.1 Pricing ANNs

The first ANN that was trained is the Black-Scholes one. The log loss plot for both the training and testing sets are given in figure 1. We can see that the BS-ANN trains well even on a rather small number of epochs.

Additionally, we plot the ANN prices against the analytical solution for a range of strikes in figure 2. The parameter values for this plot are $\tau = 1, r = 0.05, \sigma = 0.5$, which are all in-sample. We can see then that the in-sample results of the ANN are good, and that it constitutes a good approximator for the real function.

The second ANN that was trained is the Heston one. Figures 3 and 4 give the same plots for this model, with the exception that we have two Heston pricers; the first one uses the regular formula to price the European call option and the second one uses the call-put parity to price. This is due to the fact that the COS method doesn't give very stable results for OTM options. We plot the values given by the second one.

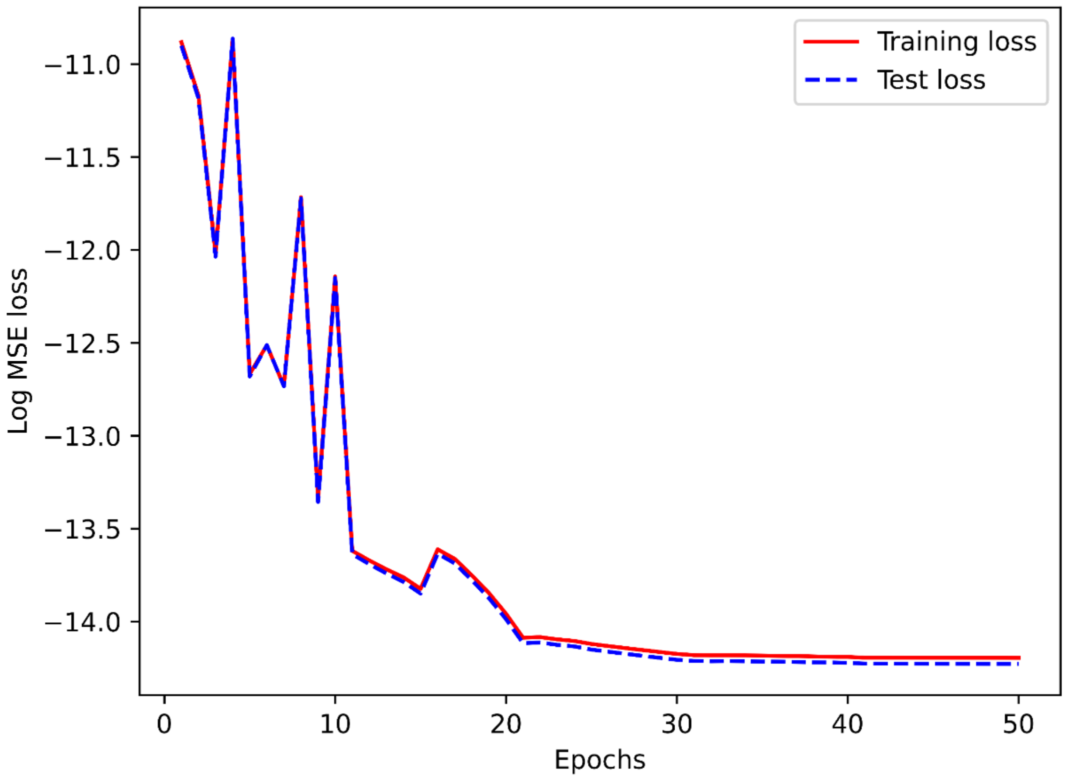

Figure 1. Black-Scholes ANN log-MSE loss

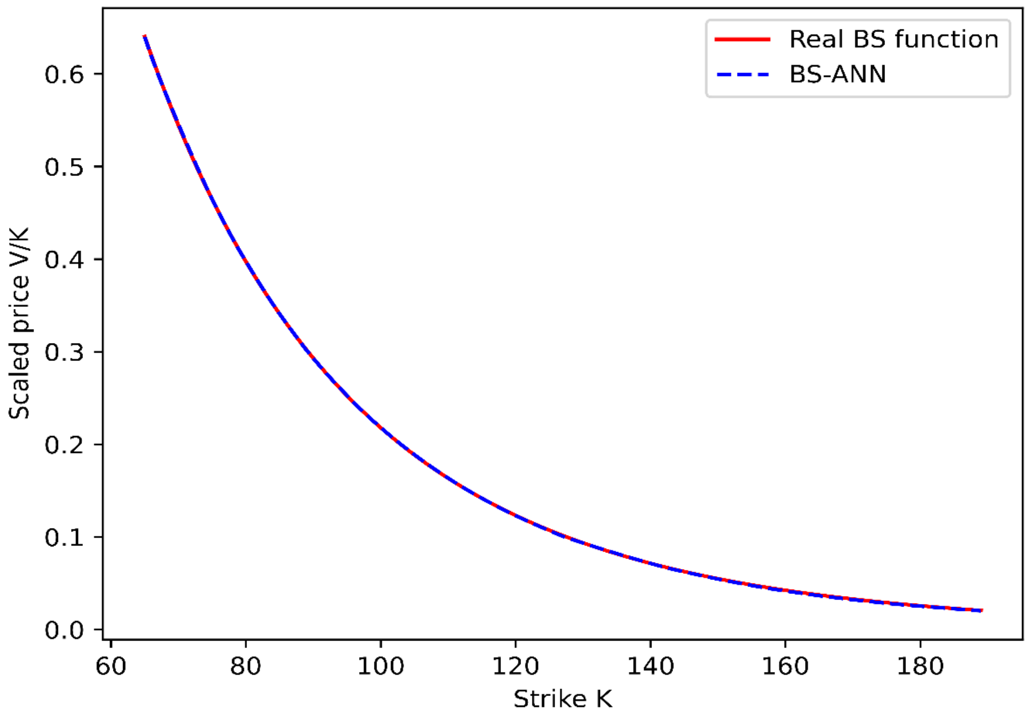

Figure 2. In-sample performance of the BS-ANN against the analytic solution

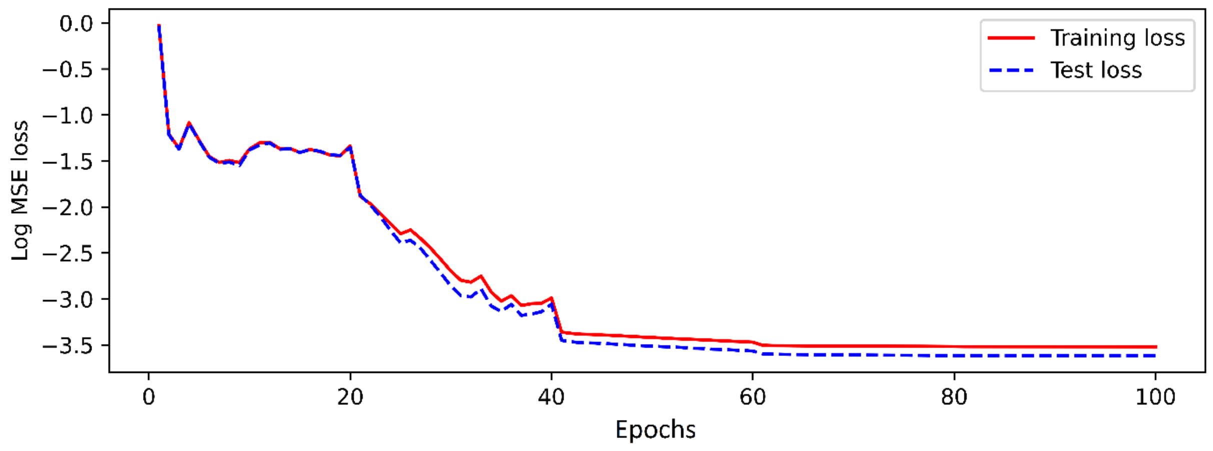

Figure 3. Heston ANN log-MSE loss

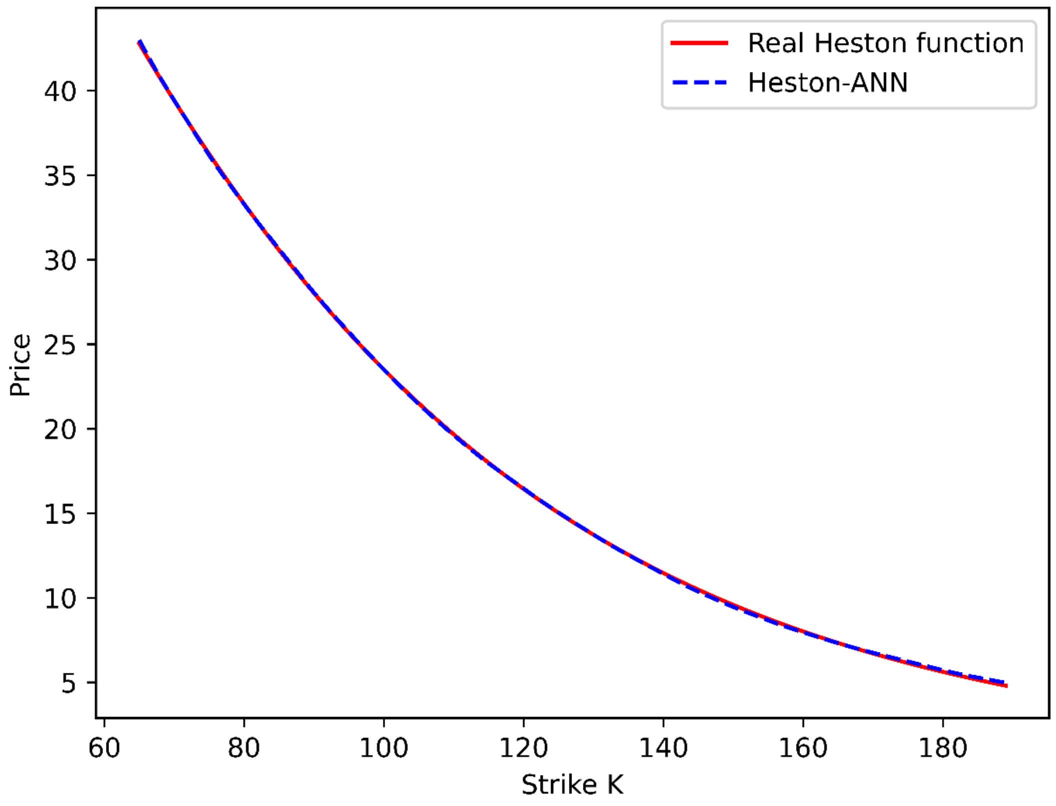

Figure 4. In-sample performance of the Heston-ANN against the analytic solution

The parameter values for the plot in figure 4 are $\tau = 1, r = 0.05, \nu_0 = 0.4, \gamma = 0.3, \kappa = 1.5, \bar{\nu} = 0.2$ and for a range of the strike. In this case as well, the in-sample performance of the ANN is good, although the loss isn't as small as we hoped for.

4.2 Implied Volatility ANNs

Although we have trained 3 ANNs that differ in small and major ways, we will only present the last trained ANN since it has the best results and to keep the report short. This ANN corresponds to the last one in table 1 and which we use to build a volatility surface, since that's the goal with this ANN. First, we give the log-loss plots for training and testing in figure 5.

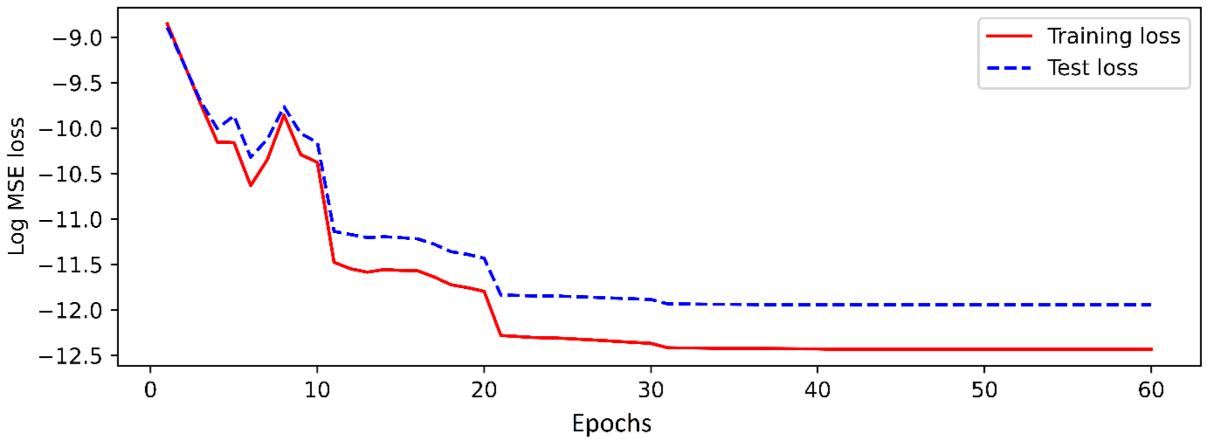

Figure 5. Implied volatility ANN log-MSE loss

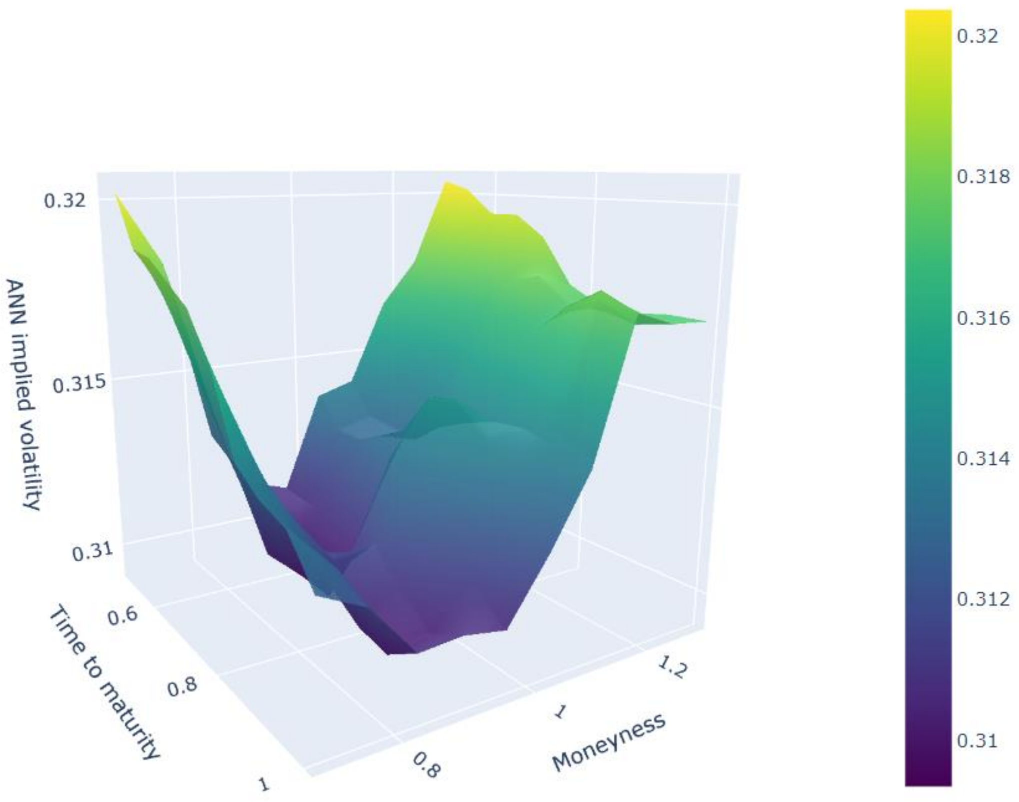

Figure 6. Volatility surface using the Heston-ANN

The parameters used to generate this volatility surface are $\rho = -0.05, \kappa = 0.3, \bar{v} = 0.1, v_0 = 0.1, r = 0.02$. As stated in the notebook, although the general shape of the curve and the implied volatility range are the best amongst all the ANNs for the implied volatility, there remains some issues with it. Mainly, the curve is not perfectly convex, as there are parts where the curve slopes inwards to create a rough looking surface.

Despite this, if we plot the heatmap of the difference between the volatility surfaces given by the ANN and Brent's algorithm, we can see that the difference is not extreme. As can be seen in figure 7, it ranges between 0.003 and -0.004, which is not extreme considering the predicted quantity, i.e. the implied volatility, ranges from 0.31 to 0.32.

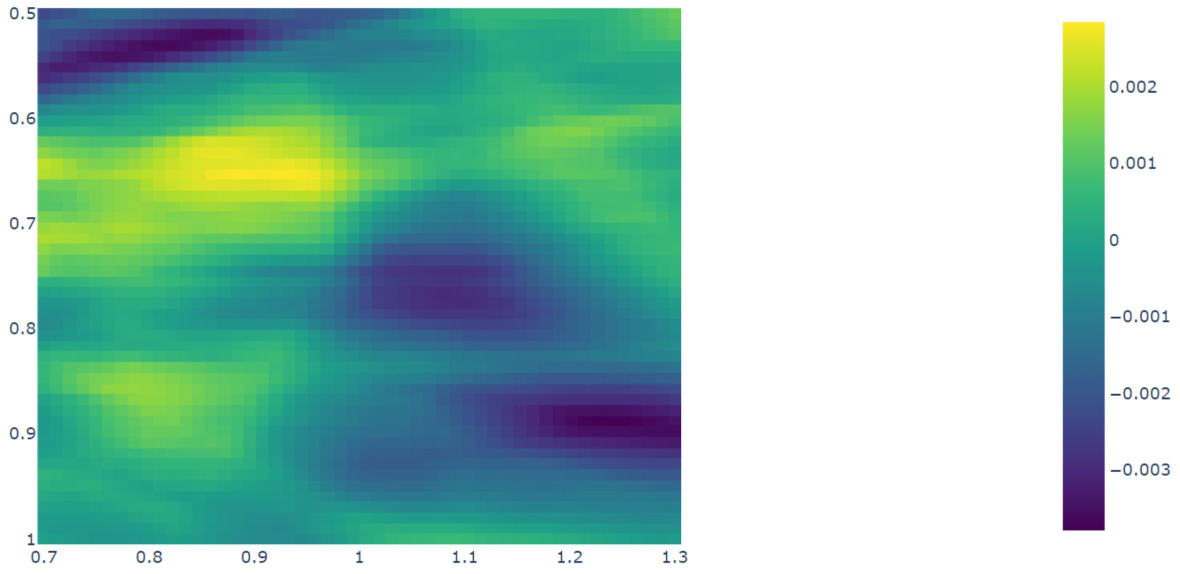

Figure 7. Heatmap of the difference between the volatility surfaces given by the ANN and Brent's algorithm

5 Discussion

In this project, we implemented a data driven approach to European option pricing and to the reconstruction of the implied volatility curve. We focused on two famous asset price models, namely the GBM and Heston models, and we used the Heston computed prices to reconstruct the implied volatility surface.

There are numerous ways this project could be taken a step further. Firstly, we could expand on the asset models to include more complicated ones, such as the rBergomi model, which has been proven to be the best when it comes to fitting real market data. In this case however, the model architecture would have to be modified to account for the non-markovian nature of this model, and therefore of the data generated by it. Additionally, we could use these models to elaborate a model hedge which we could then use to train the network, as deep is also an important question in finance.

6 Bibliography

[1] Fang, F. (2010). *The COS Method: An Efficient Fourier Method for Pricing Financial Derivatives*. (pp. 35-43)