International Journal of
Science and Engineering Investigations vol. 11, issue 126, July 2022
Received on July 23, 2022 ISSN: 2251-8843

American Option Pricing Based on Wavelet Denoised Machine Learning Methods

Avi Ray[1], Benjamin Wu[2], Julia Zhao[3], Xiaodi Wang[4]
[1]Westhill High School
[2]Horace Mann School
[3]Ridgefield High School
[4]Western Connecticut State University
([1]aviganatraray@gmail.com, [2]benwu200510@gmail.com, [3]julia.l.zhao1@gmail.com, [4]xiaodiwang1@yahoo.com)

Abstract- Financial markets are often volatile and offer high incentives to predict different option prices. Many algorithms have been proposed such as Black-Scholes model, Monte Carlo Simulation, Binomial Model, to price options. However, there are many drawbacks to each of these methods. In this research, we investigate separate methods and create a new algorithm for calculating American option pricing. To accomplish this, a variety of historical stock data was collected. Then, a subset of this data was taken, with dates ranging from 2020 to 2021, and was denoised using a 4-band wavelet transform. The data was then applied to several machine learning methods, which are Support Vector Regression (SVR) and Neural Networks; then we fitted these methods into a Least Squares Monte Carlo Simulation. The results of these tests are further compared and discussed in detail with other traditional methods.

Keywords*-Black-Scholes, 4-Band Wavelet Transform, Least Squares Monte Carlo Simulation, Neural Network, Support Vector Regression*

I. INTRODUCTION

Options are defined as contracts in which the buyer/owner of the contract can buy or sell a security at or before a specified expiry date. There are two types of options: Calls and Puts. Put options are contracts that enable the buyer to sell a security whilst a Call option enables the buyer to buy a security. The price a stock is worth is called the strike price, while the date it expires on is called the expiration date. The fee paid to purchase an option is called the premium. If the stock is below this strike price on the expiration date, the buyer loses the premium paid. However, if the stock is higher than the strike price, the profit is the difference in prices, minus the premium. Option pricing theory estimates the value of an option contract by taking the premium and calculating the probability that the contract will finish in the money.

Though the goal of finding an option pricing formula was first undertaken at the beginning of the 20th century [15], The Black Scholes Model was the first true gold standard for Option Pricing Theory and the first model to theoretically price option contracts, though with many restrictions [1]. The model was developed in 1973 by Fischer Black, Robert Merton, and Myron Scholes and is still widely used to calculate the theoretical value of an option using five input variables. These variables are the strike price of an option, the current stock price, the time to expiration, the risk-free rate, and the volatility [1]. However, the Black Scholes Model only accounts for options that cannot be exercised before the date. This is the main feature that separates American and European options. While American Options can be exercised before the agreed upon expiration date, European Options may not. This means that when handling American Options, the Black Scholes Models is not the most effective way to calculate option prices, as it completely ignores the possibility of early exercising.

After the proposal of the Black-Scholes Model, there have been many other researchers that have developed more reliable methods using different techniques. One of the most used models used by trades is the Binomial or Trinomial models. The binomial model, first developed in 1979, is a simple way to calculate option prices. At certain time periods, an asset price can move with two outcomes. The asset price can either go up a certain amount or go down a certain amount. For each time, there is a calculated percentage that the stock price will go either up or down. With this simplicity, a binomial tree may be easily modeled. A binomial tree is also more effective at calculating American Options as traders may use the predictions of the binomial tree to decide whether exercising early will, based on probability, give greater returns. However, the asset prices in the real market will almost never follow the exact values of the predicted Binomial tree. One method utilized to improve the accuracy of the Binomial tree is the Trinomial tree. The Trinomial Tree retains the option for the stock price to move up or down but adds a third option: that the asset price has no net change from one time to another [14].

Another important and one of the most popular methods to value options is the Monte Carlo Simulation. It was first developed in 1977 by Phelim Boyle to price European Options

[25]. Over time, however, many different types of Monte Carlo Simulations have been made each with their own unique feature. The Simulation is based off several random sample paths, and it uses that randomness in these paths to determine a numerical value. It presents many advantages such as being shown to be very accurate and that it can have a large range of inputs. It also presents some disadvantages, mainly being that the Simulation seems to not factor in extreme financial situations. Variations of the Monte Carlo Simulation, on the other hand, have been used to calculate additional types of alternatives as time has passed.

One of the most popular variations is the Least Squares Monte Carlo. Developed in 1996 by Jacques Carriere for options that can be exercised before the expiration date (Bermudan, American) [19]. It accomplishes this feat by a 2-step process. First, recursively, we go backward through the timestamps and assign values at each time stamp. These values will be our early exercise options. Secondly, we go through these values and decide whether to exercise the option at any of the given timestamps. This has become one of the most effective ways to price American options because of its accuracy coupled with its far lesser restrictions compared to the Black-Scholes model. However, there are some disadvantages to this method, those being that if poor constraints are set within the model the output will not be accurate, and if many input variables are used the method can become very inefficient and may crash.

To address the shortcomings mentioned above, this study utilizes Wavelet denoised pre-processing procedure combined with two different machine learning methods: Support Vector Regression (SVR) and Neural Networks, to value the premium of the option contracts The results of the machine learning methods are then analyzed and compared to the traditional models mentioned above.

II. Preliminaries

A. Wavelet Transform

In recent years, newly developed Wavelet theory has become largely popularized by the work of scientists such as Ingrid Daubechies, Ronald Coifman, and Victor Wickerhauser [26, 27]. In order to perform the wavelet denoising of the historical stock data, we must first create an M-band wavelet transform matrix that we can then use in the wavelet denoising process. Let M be a positive integer (greater than or equal to 2). An M-band wavelet transform involves the decomposition of a N dimensional signal (where N = 4^k) into M different frequency levels. The M-Band Wavelet Transform is determined by M sets of filter banks satisfying certain properties [18]. The filter bank values are able to separate the signal into the different components [17]. In our research we used 4-Band Wavelets. Let α,,, and δ denote the filter banks. Then:

$$\sum_{\kappa=1}^{8} \alpha_k = \sqrt{4} = 2$$

$$\sum_{k=1}^{8} \beta_\kappa = \sum_{k=1}^{8} \gamma_\kappa = \sum_{k=1}^{8} \delta_\kappa = 0$$

$$|\alpha| = |\beta| = |\gamma| = |\delta| = 0$$

$$\alpha.\beta = \alpha.\gamma = \alpha.\delta = 0$$

$$\beta.\gamma = \beta.\gamma = \gamma.\delta = 0$$

The structure of the M-Band wavelet transform matrix can be generalized into a matrix with dimensions $M^K \times M^K$ (where K is a positive integer) where the matrix is orthogonal. Therefore, its transpose is equal to its inverse. For the construction of a wavelet transform matrix for our experiments, we created a MATLAB function that was able to generate a wavelet transform matrix of any dimension M^K, given the band type of M and the filter bank values. In this research we generated a four band 256×256 wavelet transform matrix with the given filter banks [18].

Below is an example of a 4-Band 16×16 wavelet transform matrix.

$$\begin{bmatrix}
\alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 & \alpha_7 & \alpha_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 & \alpha_7 & \alpha_8 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \alpha_6 & \alpha_7 & \alpha_8 \\
\alpha_5 & \alpha_6 & \alpha_7 & \alpha_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 \\
\beta_1 & \beta_2 & \beta_3 & \beta_4 & \beta_5 & \beta_6 & \beta_7 & \beta_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \beta_1 & \beta_2 & \beta_3 & \beta_4 & \beta_5 & \beta_6 & \beta_7 & \beta_8 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \beta_1 & \beta_2 & \beta_3 & \beta_4 & \beta_5 & \beta_6 & \beta_7 & \beta_8 \\
\beta_5 & \beta_6 & \beta_7 & \beta_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \beta_1 & \beta_2 & \beta_3 & \beta_4 \\
\gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \gamma_5 & \gamma_6 & \gamma_7 & \gamma_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \gamma_5 & \gamma_6 & \gamma_7 & \gamma_8 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 & \gamma_5 & \gamma_6 & \gamma_7 & \gamma_8 \\
\gamma_5 & \gamma_6 & \gamma_7 & \gamma_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \gamma_1 & \gamma_2 & \gamma_3 & \gamma_4 \\
\delta_1 & \delta_2 & \delta_3 & \delta_4 & \delta_5 & \delta_6 & \delta_7 & \delta_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \delta_1 & \delta_2 & \delta_3 & \delta_4 & \delta_5 & \delta_6 & \delta_7 & \delta_8 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_1 & \delta_2 & \delta_3 & \delta_4 & \delta_5 & \delta_6 & \delta_7 & \delta_8 \\
\delta_5 & \delta_6 & \delta_7 & \delta_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \delta_1 & \delta_2 & \delta_3 & \delta_4
\end{bmatrix}$$

Figure 1. Example of a 16x16 4-Band Wavelet Transform Matrix

Below are the given filter bank values that we used for our wavelet transform matrix in the wavelet denoising process. [7]

α=[−0.067371764;0.094195111 ;0.40580489;0.567371764;0.567371764; 0.40580489; 0.094195111; −0.067371764]

β=[−0.094195111; 0.067371764; 0.567371764; 0.40580489; −0.40580489; −0.567371764; −0.067371764; 0.094195111]

γ=[−0.094195111; −0.067371764; 0.567371764; −0.40580489; −0.40580489; 0.567371764; −0.067371764; −0.094195111]

δ=[−0.067371764; −0.094195111; 0.40580489; −0.567371764; 0.567371764; −0.40580489; 0.094195111; 0.067371764]

B. Wavelet Denoising

Before subjecting the historical stock data to different tests, the data was first denoised to preserve key features. This was done by using a wavelet transform method. Historical stock data was placed into a 256×1 column vector. A 256×256 wavelet transform matrix was then created using predetermined filter banks. The stock data vector and the wavelet transfer matrix were them multiplied to create a new 256×1 column vector. The data was then split into four separate 64×1 column vectors. The first vector containing rows 1 to 64 representing low frequency components of original data was left untouched while the other three column vectors representing higher frequency components of data were denoised. A threshold value used for denoising procedure was calculated individually for each of these three high frequency components of data. To calculate the threshold value, the median absolute deviation also was calculated for each matrix with a method in MATLAB. The absolute value of each value of each high frequency component was then compared to its respective threshold values [7]. In case the value would be lower than the threshold it would be changed to zero. Otherwise, if it was greater than or equal to the threshold value, the value would remain the same. Throughout our testing, most of the values ended up being set to zero. After the threshold filtering, we recombined the first 64 rows along with the new 192 rows to make another 256×1 column vector. We multiplied that with the transpose of the wavelet transform matrix to find the denoised vector [16, 18]. This process was repeated 4 times for each respective time stamp. Let W represent the Wavelet Transform matrix and T represent the stock price vector. Then Wavelet Transform of T is given by WT= $\begin{bmatrix} a_1 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix}$ where a_1 represents the low frequency component while d_i represents the high frequency components for $i = 1,2,3$. Following is the mathematical formulation of the denoised procedure where d_{ij} represents the j^{th} coordinate of d_i, for $i = 1,2,3$.

Let λ_i be the threshold value:

$$\lambda_i = \sigma_i \sqrt{2 \log n}, n = 4^{\kappa-1}, \sigma_i = \frac{Mean\ Absolute\ Deviation\ of\ d_i}{0.6745}$$

i=1,2,3

j=1,2,3,…,$4^{\kappa-1}$

If $|d_{ij}| < \lambda_i$ then set $d_{ij} = 0$.

If $|d_{ij}| \geq \lambda_i$ then set $d_{ij} = d_{ij}$.

After threshold is applied, recombine the low frequency component with the 3 denoised high frequency components into a singular vector and multiply the vector by the transpose of the wavelet transform matrix, W, to get the denoised “clean” vector. Figures 2-5 below show the denoised stock data overlaid and compared to the historical stock data for each timestamp.

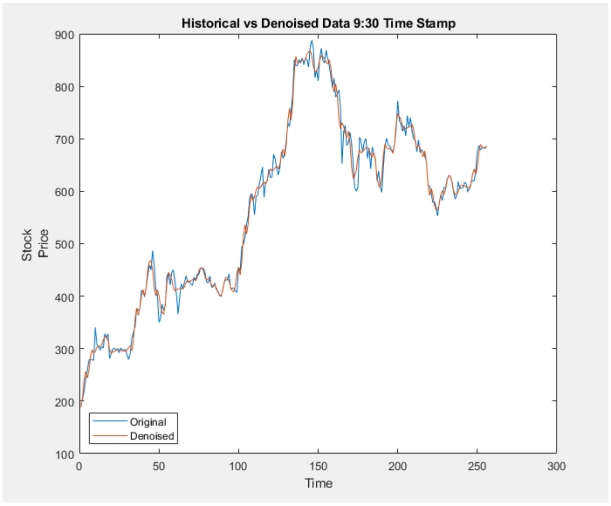

Figure 2. Historical vs. Denoised Data for 9:30

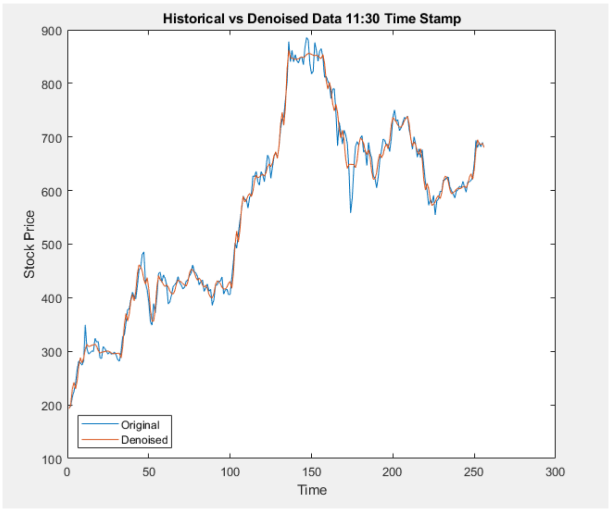

Figure 3. Historical vs. Denoised Data for 11:30

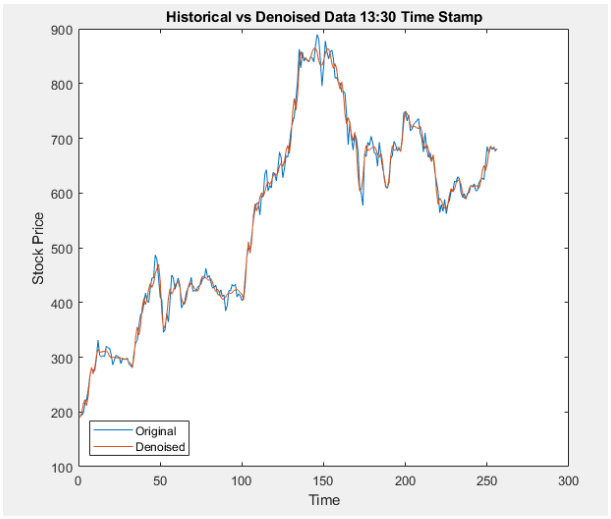

Figure 4. Historical vs. Denoised Data for 13:30

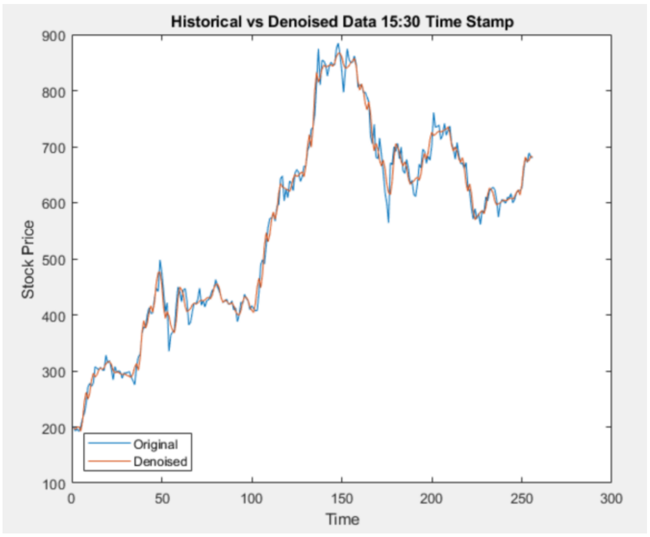

Figure 5. Historical vs. Denoised Data for 15:30

III. MACHINE LEARNING BASED METHODS

A. SVR Models

Support Vector Regression is a machine learning algorithm which uses labeled data to output singular values. SVR aims to find a hyperplane that maximizes the margin within a certain range and can be effective in both linear and non-linear examples to predict stock prices. SVR results are often quantified with a Root Square Mean Error value to show the accuracy between the predicted prices and the actual prices.

SVR uses a training set of predictors $\{x_1, x_2, \ldots\ldots, x_n\}$ and the predictor's respective label values $\{y_1, y_2, \ldots\ldots, y_n\}$. The goal of the predictors and labels is developing a model which makes predictor values within a certain error bound ϵ which allows us to choose how tolerant the bound is.

Instead of having to express our output values as binary which happens in SVM we can express it as all real numbers. Thus, knowing this we can express y as:

$$y_i = \sum_{i=1}^{n}(\alpha_i - \alpha_i{}^*) * K(x_i, x_j) + b$$

where $K(x_i, x_j)$ is a Gaussian kernel which we use to predict our stock price paths.

Of course, the predictor values cannot always be completely accurate so we can introduce slack variables ξ_i, ξ_i^* which denotes the deviations from the error bound. This transforms into an optimization problem, which can be written as:

$$min\ \frac{1}{2}||w||^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i{}^*)$$

subject to:

$$y_j - (\sum_{i=1}^{N}(\alpha_i - \alpha_i{}^*) * K(x_i, x_j) + b) \le \epsilon + \xi_j$$

$$\sum_{j=1}^{N}(\alpha_i - \alpha_i{}^*) * K(x_i, x_j) + b) - y_i \le \epsilon + \xi_j{}^*$$

where ξ_j , $\xi_j^* \ge 0,\ and\ 1 \le j \le N$. Therefore our SVR regression function is given by:

$$f(x) = \sum_{j=1}^{N}(\alpha_i - \alpha_i{}^*) * K(x, x_j) + b)$$

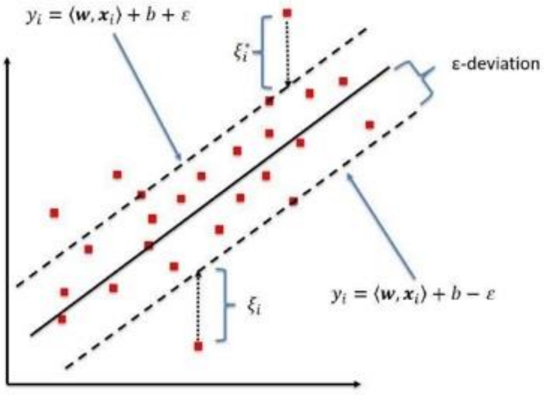
$y_i = \langle w, x_i \rangle + b + \varepsilon$
ξ_i^*
ε-deviation
ξ_i
$y_i = \langle w, x_i \rangle + b - \varepsilon$

Figure 6. Linear SVR Kernel

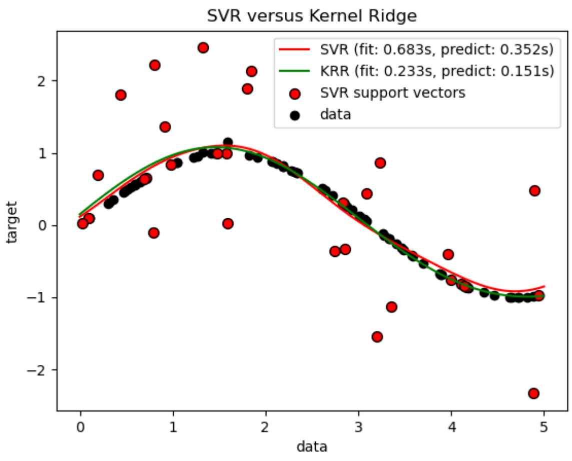

Figure 7. Curved SVR Kernel

The key values that we focused on in our SVR calculations were the Mean Square Error, and by extension, the Root Mean Square Error (RMSE), to see how much the predicted value of the stock data varied from the actual historical stock data. We utilized a regression learner toolbox in MATLAB to conduct all our calculations. We also ran regressions with several kernels, including linear, quadratic, cubic, fine Gaussian, medium Gaussian, and coarse Gaussian. We tested both our original stock data and our wavelet denoised stock data as inputs in two different experiments. Within MATLAB, we created an identical corresponding matrix for both the historical and denoised data. First, we used the historical data as an input for the different SVR models. We then compared the model against the identical copy of the historical data to see how it performed. The toolbox outputted a RMSE value which was then converted into a standardize RMSE value. Other values we took note of included the R-squared values and the Mean absolute error. This step was repeated with all the different methods of SVR previously mentioned for a singular time value. We then repeated these steps with the three other times that we chose throughout the day and then repeated all these steps for our denoised data. Our output for the RMSE values and R-squared values was evaluated by taking the average of the 4 different outputs at each respective time stamp (9:30, 11:30, 13:30, 15:30) for each separate kernel. (Done separately for both the historical and denoised data).

1) *Historical Data*

TABLE I. RMSE AND R-SQUARED VALUES FOR EACH SVR KERNEL TYPE

Kernel	RMSE Values	R-Squared Values
Linear	10.021	1.00
Quadratic	13.000	.99
Cubic	15.815	.99
Fine Gaussian	29.151	.97
Medium Gaussian	18.364	.99
Coarse Gaussian	15.110	.99

2) *Denoised Data*

TABLE II. RMSE AND R-SQUARED VALUES FOR EACH SVR KERNEL TYPE

Kernel	RMSE Values	R-Squared Values
Linear	10.072	1.00
Quadratic	12.889	.99
Cubic	15.234	.99
Fine Gaussian	20.790	.99
Medium Gaussian	15.632	.99
Coarse Gaussian	11.994	1.00

3) *Discussion of Tables I and II*

As you can see the table that contains the cleansed/denoised data has overall smaller RMSE values. This illustrates how with the denoising process the data becomes more accurate, as RMSE illustrates the amount of error the prediction has made and comparing the RMSE values of the denoised table with those of the true historical data we can see that the SVR makes more accurate predictions using the denoised data as compared to the historical data. Also comparing the R-Squared values of each table we can see how the denoised data has a higher R-Squared value on average as compared to the true historical data. This means the correlation between the SVR predicted stock price values and the real stock price values is higher using the denoised data as compared to using the true historical data.

B. Neural Networks

Neural network is an artificial intelligence-like machine learning technology that simulates the working principle of the human brain. It builds a mathematical model or calculation model that loosely mimics the structure and function of a biological neural network (such as an animal's central nervous system, especially the brain), and is used to estimate or approximate functions. They support the processing of multiple data types such as images, texts, voices, sequences, etc., and can realize classification, regression, and prediction.

There are many types of neural networks that have evolved. A common Multilayer Network consists of the following three parts. These three parts are visualized in Figure 8 below.

1. Input layer: A multitude of neurons accept many non-linear input information. The input message is called the input vector, which is the input data.
2. Hidden layer(s): composed of many neurons and links between the input layers themselves by activation functions and the output layer, responsible for data processing. Although hidden layers can have numerous layers, they are usually only employed with one. The number of nodes (neurons) in the hidden layer is variable, but the larger the number, the more significant the non-linearity of the neural network, and the more obvious the robustness of the neural network.
3. Output layer: in the output layer, information is exchanged, evaluated, and weighed to provide an output result. The output message is called the output vector.

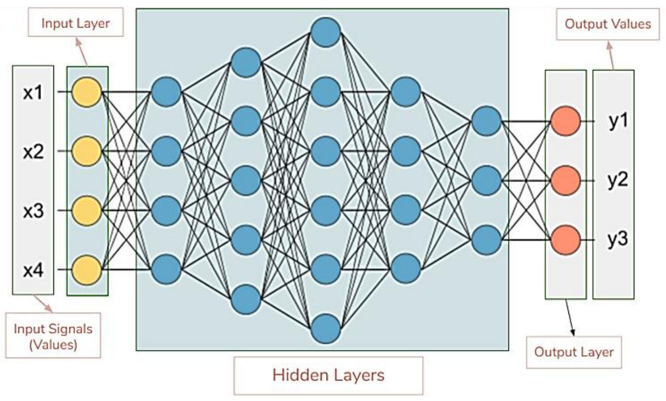

Figure 8. Model of a standard Neural Network illustrating three different layers

Layers contain nodes each with values. Usually, these nodes are connected to all the other nodes in the next and previous layer, and the connection that is calculated based on training data. A node's value is calculated as the summation of the weight multiplied by the node value of all the nodes is inputted into an activation function, and the resultant value is transmitted to the next layer's appropriate nodes. This is calculated throughout all the hidden layers until finally the nodes in the output layer have been filled, and then the network generates results. To adjust the neural network, the "connection weight" between neurons and the threshold of each functional neuron must be adjusted according to the training given samples using backward propagation gradient descent method. The final model is expected to have a certain generalization ability for unknown samples.

The number of hidden layer neurons is highly crucial to determine throughout the network design process. Too many hidden layer neurons will increase the number of network computations and easily create over-fitting issues; too few neurons will impair network performance and lead it to fail to reach the desired outcomes. The number of hidden layer neurons in a network is proportional to the difficulty of the issue, the number of input and output layers, and the error thresholds.

The neural network algorithm can build a non-linear model driven by market data by simulating the neuron algorithm and obtain a better pricing effect than the parameter model, which makes option pricing more objective and accurate.

We used the MATLAB Neural Network Time series toolbox to predict the stock prices, for both our denoised as well as real data. RMSE values and the error graphs were recorded and presented below.

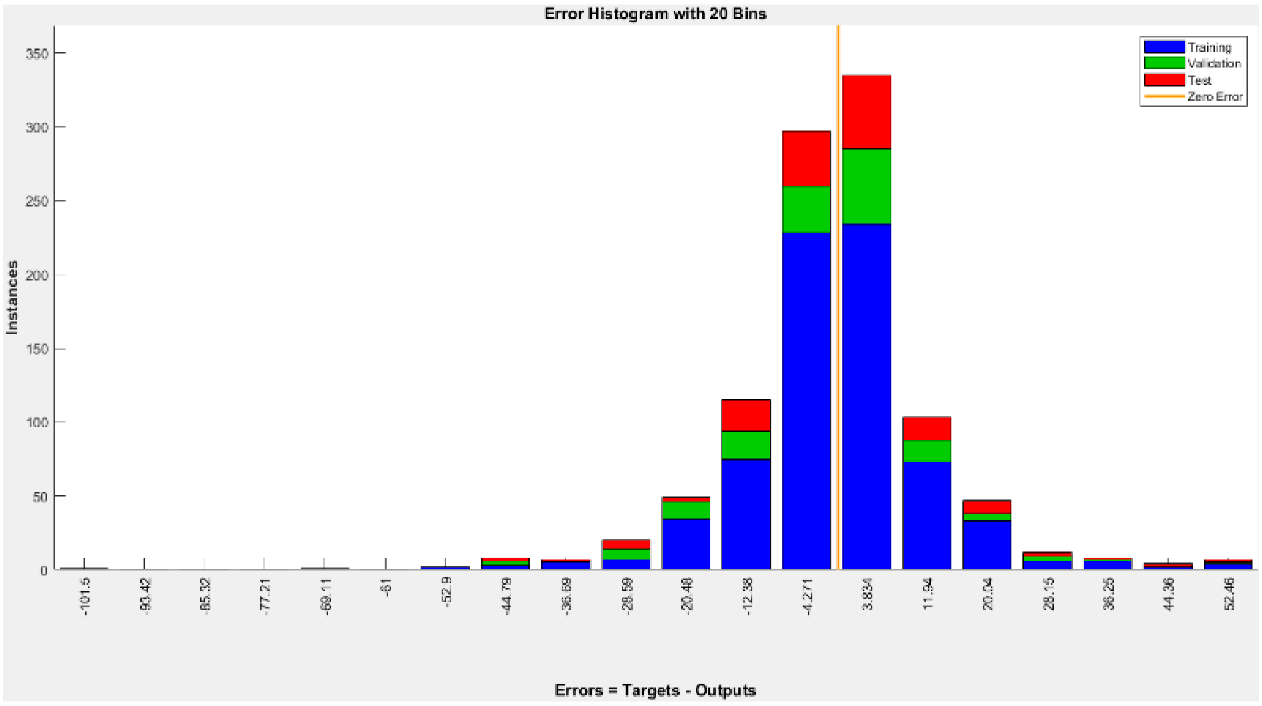

Figure 9. Historical Stock data. RMSE:14.9

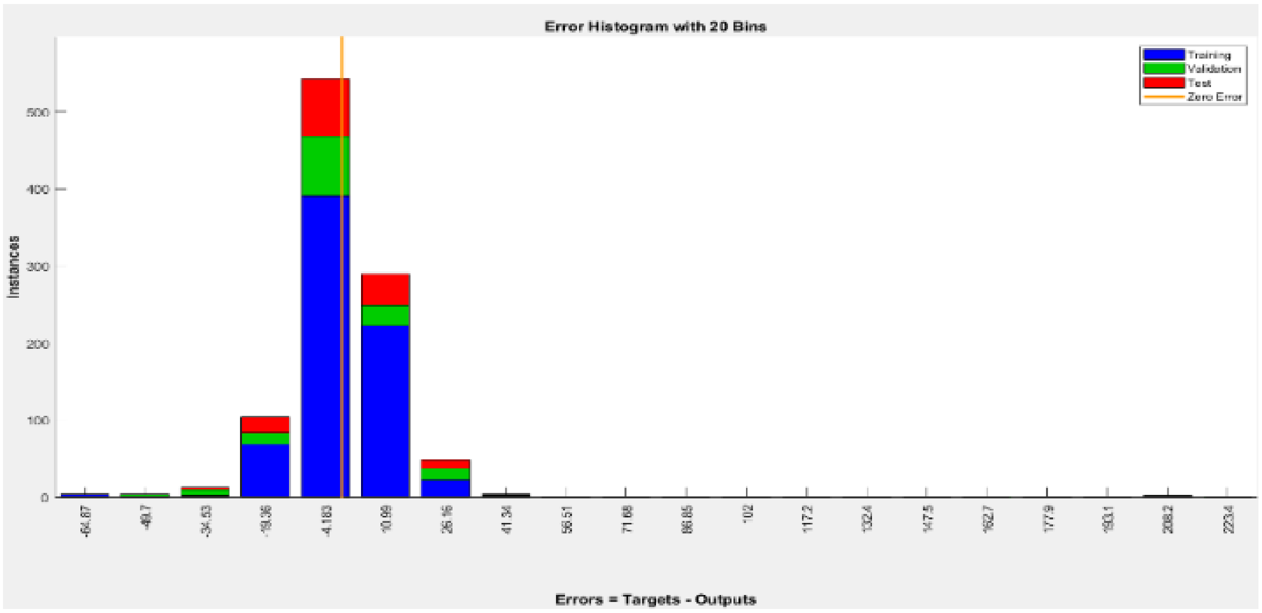

Figure 10. Denoised Stock Data. RMSE: 13.7

1) Discussion of Figures 9 and 10:

As shown in the 2 figures above, the graph illustrating the denoised data has the lower RMSE value indicated compared to the Historical Data, meaning that the neural network was able to better predict the stock price data using the denoised data as compared to the Historical Stock Price Data.

C. Monte Carlo Simulation

Introduced by Boyle in the late 70s the Monte Carlo Simulation has been one of the main methods to evaluate the value of an option [25]. The Simulation is based on risk neutral valuation (Meaning we can assume that all assets will grow and be discounted at a risk-free rate). The Simulation itself is in 3 critical steps:

1. Many random paths are generated of the underlying variables
2. Calculate the “payoff” value of the option in each path created in step 1
3. Values are discounted to today and averaged

The result is the price of the option.

Mathematically the option price *O* in the Monte Carlo Simulation can be written as an integral that illustrates the expected value of the discounted payoff under a risk neutral probability measure. [13]

$$[0,1)^t = \{u = (u_0, \ldots\ldots, u_{t-1}) : 0 < u_j < 1 \; for\ all\ j\}:$$

$$O = O(f) = \int_0^1 \ldots\ldots \int_0^1 f(u_0, \ldots\ldots, u_{t-1}) du_0 \ldots\ldots du_{t-1} = \int f(u)du = E[f(U)]$$

For some function $f : [0,1)^t \rightarrow \mathbb{R}$ and where u represents a point in $[0,1)^t$, and U~U$[0,1)^t$ is a random point with uniform distribution.

1) Least Squares Monte Carlo Method:

The Least Squares Monte Carlo Method is one of the methods for evaluating American Style choices. The approach use the N random route created by geometric Brownian motion for stock price prediction. The random paths can be expressed as: $(S_n^k, t_n)\, for\ 1 \leq k \leq N$ and $t_n = ndt$. If we know $F_{n+1}^k = F(S_{n+1}^k, t_{n+1})$, we can set $X = S_n^k$ which is the current equity value and $Y = e^{-rdt}F(S_{n+1}^k, t_{n+1})$ which represents the value of the deferred exercise. Next, we can perform the regression of Y as a function of polynomials $X, X^2, X^3, \ldots\ldots, X^m$. We can use this regressed value to decide whether to exercise early. Meaning that option holders will compare whether the immediate payoff value is higher than the projected payoff value with continuation. If the immediate payoff value is higher the option holder will exercise the option, if it is not higher then they will not exercise the option. [13]

2) Wavelet Denoised Based Monte Carlo Method.

Using the Support Vector Regression and the Neural Network Methods mentioned before we inputted the historical stock data into each of the respective MATLAB toolboxes. The prediction paths from each of these methods would each become the paths for our Monte Carlo Method. Then using each of these paths we manually performed the least squares method as mentioned above. For the Least Squares Monte Carlo, we used the prebuilt MATLAB method in the financial toolbox.

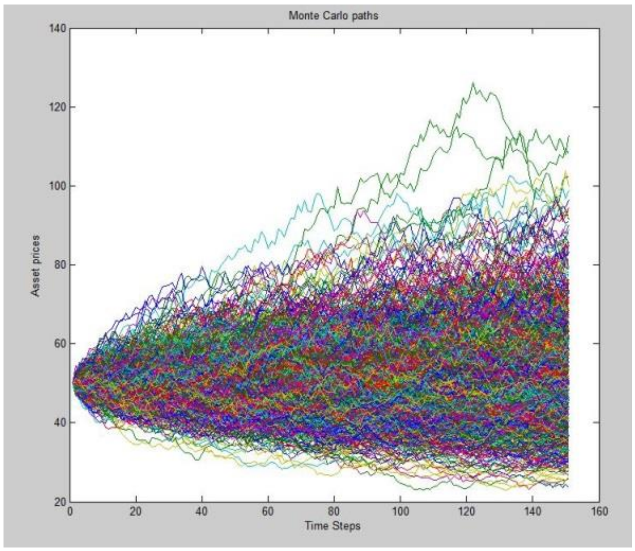

Figure 11. A graph of the Monte Carlo Simulation

IV. Experimental Results

A. Input Data (Put Option)

TABLE III. Input data of three different Tesla options selected from www.yahoo.com

Strike Price	Volatility	Initial Strike Price	Expiration Date
350	141.41%	711.92	8/27/2021
400	110.55%	711.92	8/27/2021
450	97.46%	711.92	8/27/2021

B. Output Data (Put Option):

TABLE IV. RMSE and Option Value results when using all four different methods

Methods	Support Vector Regression	Neural Networks	Brownian Motion	Black-Scholes
RMSE	0.054	0.0009	0.026	0.034
Option Value 1.	0.06366	0.023014	0.0535	0.0348
Option Value 2.	0.21921	0.1614	0.1719	0.1323
Option Value 3.	0.35812	0.3198	0.3091	0.2642

V. Discussion

For our experiments, our goal was to test which of the methods was the most accurate in predicting option prices. To do this, for each method, we calculated the previously mentioned RMSE (Root-mean-squared error). The RMSE is a way to quantify the error between the values predicted by a model and the values observed. The RMSE was calculated in

MatLab toolboxes that we also used to calculate option prices. The goal is to minimize the RMSE value to prove a high degree of accuracy between option prices the model predicted and the observed values.

Between the four methods, Support Vector Regression, Neural Networks, Monte Carlo Method, and Black-Scholes, Neural Networks inputted into a Least Squares Monte Carlo Method had the lowest RMSE score with a score of 0.0009. The Black-Scholes model had a higher RMSE value of 0.034 which is expected. As mentioned above, this model has many limitations, and doesn't consider the unique nature of American Options. Interestingly, the SVR had the highest RMSE value of 0.054. This may be due to the variety of different kernels that we used for these calculations. In the future, we can look for a specific kernel that is the most effective for this problem and use that. Finally, the Brownian Motion based Monte Carlo method had an RMSE value of 0.026 which was the second lowest out of the four methods.

However, all four of these RMSE values were very low (standardized RMSE values range from 0-1). Hence, all the models were still quite accurate when compared to the observed values. To further this research in the future, other methods that we mentioned but did not test can be considered. As mentioned earlier, a specific SVR kernel should be determined, and Neural Network calculations can also be specific to become an extremely accurate predictor.

VI. Conclusion Remarks and Future Research

In this paper, the put options are predicted based on the stock price, the exercise price, the time to expiration, the interest rate, and the volatility. Firstly, we utilized a 4-Band Wavelet Transform to clean up the data and make it more reliable. After applying Support Vector Regression and Neural Networks to wavelet based denoised stock price data, we were able to predict the stock price path at a high accuracy and then we inputted these paths into a Least Squares Monte Carlo Function. We then compared these results to the more traditional methods such as Black-Scholes and the Least Squares Monte Carlo using Brownian Motion. The experiments and comparisons have determined that the Neural Network method is the best fit and is more accurate than the other methods. As for future research to make the algorithm more in depth, Wavelet based Quasi Monte Carlo Methods could be used, a Wavelet based binomial tree method also could be added, and more in-depth SVR and Neural Networks could be appended for more precise predictions.

Appendix For Code

Original code to create the wavelet transform matrix:

m, being a matrix of 0s, variables being a matrix of the filter banks.

```
function [m] = filling(m,bandtype,variables)
i = 1;
count = 0;
c = 1; % colum in variables
len1 = length(m);
len2 = length(variables);
x = len2/2;
for l = 1:len1
  for j = 1:bandtype
    if i + len2 -1 > len1
        m(l,1:x) = variables(c,x+1:len2);
        m(l,(len1-x)+1:len1) = variables(c,1:x);
    else
    m(l,i:i+len2-1)= variables(c,:);
    end
  end

  i = i + bandtype;
  count  = count +1;
  if count == length(m)/bandtype
    count = 0;
    i = 1;
    if c == length(m)/bandtype
        c = 1;
    else
        c = c + 1;
    end
  end
end
end
```

Original code to calculate the threshold values and denoise the historical stock data vector:

T, being the original historical stock data vector.

```
A1 = A(1:64,:);
D1 = A(65:128,:);
D2 = A(129:192,:);
D3 = A(193:256,:);
[thresholdsD1, thresholdsD2,
thresholdsD3]=thresholds(D1,D2,D3)

D1a=filteredD1(D1,thresholdsD1)
D2a=filteredD2(D2,thresholdsD2)
D3a=filteredD3(D3,thresholdsD3)
E=[A1;D1a;D2a;D3a]
X=transpose(S)*E

function [thresholdsD1,thresholdsD2,thresholdsD3] =
thresholds(x,y,z)
B=mad(x)
thresholdsD1=(B/0.6745)*sqrt(2*log(256))
C=mad(y)
thresholdsD2=(C/0.6745)*sqrt(2*log(256))
D=mad(z)
thresholdsD3=(D/0.6745)*sqrt(2*log(256）)
end
 function [D1a] = filteredD1(x,y)
counter=1
while counter <= 64
```

```
        if abs(x(counter,1))>=y
            counter=counter+1;
        else
            x(counter,1)=0;
            counter=counter+1;
        end
    end
    D1a=x
    end

    function [D2a] = filteredD2(x,y)
    counter=1
    while counter <= 64
        if abs(x(counter,1))>=y
            counter=counter+1;
        else
            x(counter,1)=0;
            counter=counter+1;
        end
    end
    D2a=x
    end

    function [D3a] = filteredD3(x,y)
    counter=1
    while counter <= 64
        if abs(x(counter,1))>=y
            counter=counter+1;
        else
            x(counter,1)=0;
            counter=counter+1;
        end
    end
    D3a=x
    end
```

ACKNOWLEDGMENTS

We would like to thank Dr. Xiaodi Wang from Western Connecticut State University. Without his constant encouragement and advice, it would have been impossible to finish this project.

REFERENCES

[1] Fisher, B., Myron, S. (2008). The Pricing of Options and Corporate Liabilities. Chicago Journals.

[2] Amit, D. (2011). Option Pricing using Machine Learning Techniques. Indian Institute of Technology, Bombay.

[3] Raquel, M. G., Sara, D. L., Bernardo, S. (2020). Neural Network Pricing of American Put Options. Rem Research Center.

[4] Alexander, K., Andrew, Y. (2019). Option Pricing with Deep Learning, Department of Computer Science, Stanford University.

[5] Shuaiqiang, L., Cornelius, W. O., Sander, M. B. (2018). Pricing Options and Computing Implied Volatilities using Neural Networks. Centrum Wiskunde & Informatica, Science Park 123.

[6] Michael, P., Tshilidzi, M. (2004). American Option Pricing Using Multi-Layer Perceptron and Support Vector Machine. University of Witwatersrand.

[7] Yinqi, C., Hieu, N., Srihita, M., Mingyang, Z., (2019). Option Pricing by Wavelet Filtering and Machine Learning Based on Monte Carlo Method.

[8] Massimiliano, F. (2013). Pricing American Options with Least Squares Monte Carlo on GPUS. NVIDIA Corporation.

[9] Stathis, T., Chunyu, Y. (2004). Pricing American-Style Options by Monte Carlo Simulation: Alternatives to Ordinary Least Squares.

[10] Ehsan, U., Fayaz, Ali., Tawfiqullah, Q. (2016). Monte-Carlo Simulation for American Options.

[11] Russel, E. C., Suneal, C. (2005). Monte Carlo Simulation for American Options. Mathematics Department, UCLA.

[12] Francis, A. L., Eduardo, S. S. (2001). Valuing American Options by Simulation: A Simple Least-Squares Approach. The Society for Financial Studies.

[13] Quiyi, J. (2009). Pricing American Options using Monte Carlo Methods. Uppsala University.

[14] Don, M. C. (2015). A Synthesis of Binomial Option Pricing Models for Lognormally Distributed Assets. Department of Finance, Louisiana State University.

[15] Reaz, C., M.R.C., M., Tanisha, N. A., Colam, D. A. Q. (). Predicting the Stock Price of Frontier Markets Using Modified Black-Scholes Option Pricing Model and Machine Learning. North South University and University of Dhaka, Bangladesh.

[16] Hieu, Q. N., Abdul, H. R. (2019). Stock Forecasting using M-Band Wavelet-Based SVR and RNN-LSTMs Models. Western Connecticut State University.

[17] Choi, K., Lee, T. (2019). Differentially Private M-band Wavelet-Based Mechanisms in Machine Learning Environments. Western Connecticut State University.

[18] Lin, T., Xu, S., Shi, Q., Hao, P,. (2006). An Algebraic Construction of Orthonormal M-band Wavelets with Perfect Reconstruction. Peking University.

[19] Carriere, Jacques F. 1996. Valuation of the early-exercise price for options using simulations and nonparametric regression. Insurance: Mathematics & Economics 19: 19–30.

[20] R. Cohen. (2012). Signal denoising using Wavelets. Project Report, Department of Electrical Engineering Technion, Israel Institute of Technology, Haifa.

[21] T.-J. Hsieh, H.-F. Hsiao, W.-C. Yeh. (2011). Forecasting stock markets using Wavelet Transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm. Applied Soft Computing, vol. 11, no. 2, pp. 2510-2525.

[22] Z. Pan, X. Wang. (1998). A Stochastic Nonlinear Regression Estimation Using Wavelets. Computational Economics, vol.11, pp. 89-102.

[23] Z. Pan, X. Wang. (1998). A Wavelet-Based Nonparametric Estimator of the Variance Function. Computational Economics

[24] Hieu Q. N., Xiaodi W. (2016). Pseudo Quantum Steganography with "Color Barcode" in M-band Wavelet Domain. International Journal of Signal Processing, 1, 150-168.

[25] Phelim P. Boyle. (1977). Options: A Monte Carlo approach, Journal of Financial Economics, Volume 4, Issue 3

[26] Daubechies, Ingrid. (1993). Wavelet transforms and orthonormal wavelet bases. Proc. Sympos. Appl. Math., 47, Amer. Math. Soc., Providence, RI

[27] Coifman, Ronald R., Wickerhauser, M. Victor. (1993). Wavelets and adapted waveform analysis. A toolkit for signal processing and numerical analysis. Proc. Sympos. Appl. Math., 47, Amer. Math. Soc., Providence, RI

How to Cite this Article:

Ray, A., Wu, B., Zhao, J. & Wang, X. (2022). American Option Pricing Based on Wavelet Denoised Machine Learning Methods. International Journal of Science and Engineering Investigations (IJSEI), 11(126), 24-32. http://www.ijsei.com/papers/ijsei-1112622-04.pdf