Proceedings of the 2021 Winter Simulation Conference
S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo, and M. Loper, eds.

OPTION PRICING BY NEURAL STOCHASTIC DIFFERENTIAL EQUATIONS: A SIMULATION-OPTIMIZATION APPROACH

Shoudao Wang

School of Data Science
Fudan University
440 Handan Road
Shanghai 200433, CHINA

L. Jeff Hong

School of Data Science and School of Management
Fudan University
440 Handan Road
Shanghai 200433, CHINA

ABSTRACT

Classical option pricing models rely on prior assumptions made on the dynamics of the underlying assets. While empirical evidence showed that these models may partially explain the option prices, their performance may be poor when the actual situation deviates from the assumptions. Neural network models are capable of learning the underlying relationship through the data. However, to avoid over-fitting, these models require massive amount of data, which are not available for option pricing problems. We propose a new model by integrating neural networks to a classical option pricing model, thus increasing the model flexibility while requiring a reasonable amount of data. We show that the training of the model, also known as the calibration, may be formulated into a simulation optimization problem, and it may be solved in a way that is compatible to the training of neural networks. Preliminary numerical results show that our approach appears to work well.

1 INTRODUCTION

Option pricing has been an important topic in the field of financial engineering for many years. The most famous work in this area is Black-Scholes model (Black and Scholes 1973; Metron 1973). A major assumption of this model is that the underlying asset price follows a geometric Brownian motion. However, it is well known that Black-Scholes model cannot capture some important phenomena in the real market, such as the heavy tail of return distributions (Cont and Tankov 2004). To capture these phenomena, different models have been proposed. Among them, two types of models are most popular: (1) Time-varying volatility models that can capture extreme values by allowing the variance of the return distribution to change stochastically over time, e.g., the well-known Heston model of Heston (1993) and the GARCH model of Heston and Nandi (2000). (2) Jump models with discontinuous jumps as a part of the underlying assets dynamics, e.g., Merton model (Merton 1976), Kou model (Kou 2002) and variance-gamma model of (Madan et al. 1998). While these models are capable of capturing certain market phenomena that were overlooked by Black-Scholes model, they are still quite rigid in fitting the data and may suffer from other problems (Bates 2003).

Recently, the fast development of machine learning techniques has enabled breakthroughs in various scientific fields, making it possible to solve many complex problems. For instance, Hornik et al. (1989) proved that any function can be approximated by a neural network with nonlinear activation functions. Therefore, by directly learning through the data, neural network models may capture complex functional relationships, and they are often more flexible compared to alternative models. Although neural network models have been used widely, they are typically data expensive, i.e., they live with the so-called "big data". When the amount of data is not enough or unevenly distributed, these models can be easily over-fitted and

978-1-6654-3311-2/21/$31.00 ©2021 IEEE

produce unstable results. So their applications are often limited to areas with massive accessible data, e.g., image processing and natural language processing.

While the data requirement may not be a problem in certain applications, in many other applications, such as option pricing, the limited data size has made it impossible to rely entirely on neural network models. In these applications, structural models with certain assumptions based on scientific literature and empirical evidence are still predominant. However, as we mentioned earlier, these structured models are often very rigid. Therefore, it is natural to ask whether one can combine the rigid structural models with more flexible neural network models. It has recently been discovered that the combination of differential equations and neural networks appears to work well in some areas. For instance, Raissi et al. (2019) proposed physics-informed neural networks (PINNs) that integrate prior scientific models by employing partial differential equations in the loss functions of neural networks, and Chen et al. (2018) use the structure of ordinary differential equations to derive an effective neural network architecture.

The purpose of this paper is to bridge the gap between traditional differential equation based option pricing models and neural network models by adding neural networks as components to the stochastic differential equations (SDEs). Take the general Black-Scholes model as an example. It assumes that the underlying asset follows a geometric Brownian motion with a constant drift and a constant volatility. However, the asset price dynamics may be much more complex. Therefore, we may keep the general structure of the SDE but substituting the constant drift and volatility terms by two neural networks that may be trained (or calibrated in the language of option pricing) using the data of option prices. We call this model a neural SDE (NSDE) model. By this way, the NSDE model integrates the general structure of diffusion models, which have been tested widely in the literature, and the neural network models, which are flexible in capturing any functional relationships. The use of Black-Scholes model is only an example. In this paper we adopt a stochastic volatility model as our base model and substitute the drift and volatility terms by neural networks.

The calibration of the option pricing models is an important problem. Classical models, for example, the Black-Scholes model and Heston's stochastic volatility model, often produce closed-form expressions of option prices. One can solve a nonlinear least squares problem, which minimizes the total squared differences between the model predicted prices and actual prices, to calibrate the model parameters. For NSDE models, however, with complex neural networks as components of the SDEs, it is impossible to derived closed-form expressions. Therefore, we propose to use Monte Carlo simulation to estimate the model predicted prices, and the calibration problem becomes a simulation optimization (SO) problem. In this paper we show how the SO problems may be naturally integrated into the standard training (i.e., calibration) of neural networks.

The rest of this paper is organized as follows. In Section 2, we introduce the NSDE formulation of the option pricing problem. Calibration of the NSDE models and option pricing using the model are studied in Section 3. Section 4 presents some preliminary numerical results, followed by a conclusion and future prospects in Section 5.

2 PROBLEM FORMULATION

In this paper we only consider European call options, and other types of options may be modeled and analyzed in a similar way. The price of an European call option at any time 0 is

$$P = \mathrm{E}\left[e^{-r_f T}\left(S_T - K\right)^+\right], \tag{1}$$

where S_t is the price of the underlying asset at time t, r_f is the risk-free interest rate, T is the expiration date, K is the strike price, $a^+ = \max(a,0)$, and the expectation is taken with respect to the risk-neutral measure. Let $P(\vec{\theta},\vec{w})$ denote the option price, where $\vec{\theta}$ denotes the vector of observable parameters (e.g., S_0, r_f, K, T etc.) and $\vec{w}$ be the vector of unobservable parameters (e.g., σ in Black-Scholes model). To use the option price $P(\vec{\theta},\vec{w})$, one has to determine the values of $\vec{w}$, which is known as calibration. Let

$(\vec{\theta}_i, P_i), i = 1, \ldots, N$, denote the historical data of option prices P_i under different observed market conditions $\vec{\theta}_i$. The calibration problem typically solves the following least squares problem

$$\min_{\vec{w}} \sum_{i=1}^{N} \left[P_i - P(\vec{\theta}_i, \vec{w}) \right]^2 \tag{2}$$

to find the optimal $\vec{w}$. Problem (2) basically tries to find the set of unobservable parameters that minimize the sum of squared differences between the model predicted prices and the observed prices.

We use Heston model as an example. To solve the problem that Black-Scholes model cannot capture extreme values observed in the markets, Heston model allows the volatility to stochastically change over time and models it by a separate SDE,

$$\begin{aligned} dS_t &= r_f S_t dt + \sqrt{V_t} S_t dB_{1t} \\ dV_t &= \kappa(\theta - V_t)\, dt + \sigma \sqrt{V_t} dB_{2t}, \end{aligned} \tag{3}$$

where r_f is the risk free rate, $\sqrt{V_t}$ is the volatility of the asset price at time t, σ and θ denote the volatility and long time mean of V_t, respectively, κ measures V_t's rate of reversion to θ. Furthermore, the two Brownian motions B_{1t} and B_{2t} are correlated with a correlation coefficient ρ, i.e.,

$$dB_{1t} dB_{2t} = \rho dt.$$

The time-varying volatility in Heston model captures two important phenomena in the markets. First, large movements of asset price are clustered. Second, decreases of asset prices often lead to increases in the volatility. The first is due to the continuity of the volatility in time, and the second is solved by introducing negative correlation ($\rho < 0$). Based on Heston model, the closed-form expression of $P(\vec{\theta}, \vec{w})$ may be derived and the calibration problem may be solved using nonlinear optimization tools.

Although Heston model is more flexible than Black-Scholes model, it still imposes restrictive assumptions on the price dynamics of the underlying asset. However, the real market relationship may be more complex. By integrating neural networks in the SDEs, we may keep the structure of the SDEs, but providing more flexibility in modeling. In this paper we propose the following NSDE model based on Heston model:

$$\begin{aligned} dS_t &= NN_1(S_t, V_t, r_f, t) dt + NN_2(S_t, V_t, r_f, t) dB_{1t} \\ dV_t &= NN_3(S_t, V_t, r_f, t) dt + NN_4(S_t, V_t, r_f, t) dB_{2t}, \end{aligned} \tag{4}$$

where NN_1,NN_2,NN_3 and NN_4 are four neural networks with different structures and weights, and B_{1t} and B_{2t} also have a correlation ρ as in the classical Heston model. We choose Heston model as our base model for two reasons: First, stochastic volatility models play a role with first-degree importance in terms of revising Black-Scholes model, and second, stochastic volatility models provide excellent hedging performance (Bakshi et al. 1997).

With the neural networks in the NSDE model, it may be impossible to derive a close-form pricing formula. Therefore, we suggest to use Monte Carlo simulation method to estimate the option price. By simulating multiple sample paths of the underlying price dynamics S_t^k, $0 \le t \le T$ for $k = 1, \ldots, n$, the option price may be estimated by the average discounted payoff of these paths at the expiration date T, i.e.,

$$\bar{P} = \frac{1}{n} \sum_{k=1}^{n} e^{-r_f T} \left(S_T^k - K \right)^+. \tag{5}$$

To simulate the sample path, which is continuous, we adopt Euler's scheme of discretization (Strikwerda 1989) and use the following recursive equations to generate S_t at m evenly distributed discrete time points

$0 = t_0 < t_1 < \cdots < t_m = T$ and $t_{j+1} - t_j = \Delta t = T/m$.The discrete version of Equation (4) using forward Euler's scheme can be formulated as Equation (6),

$$\begin{aligned} S_{t_{j+1}} &= S_{t_j}(1 + NN_1(S_{t_j}, V_{t_j}, r_f, t_j)\Delta t + NN_2(S_{t_j}, V_{t_j}, r_f, t_j)\sqrt{\Delta t} * Z^1_{t_j-1}) \\ V_{t_{j+1}} &= V_{t_j}(1 + NN_3(S_{t_j}, V_{t_j}, r_f, t_j)\Delta t + NN_4(S_{t_j}, V_{t_j}, r_f, t_j)\sqrt{\Delta t} * Z^2_{t_j-1}), \end{aligned} \tag{6}$$

where $Z^1_{t_j-1}$ and $Z^2_{t_j-1}$ are two standard normal random variables with correlation ρ, and S_{t_j} and V_{t_j} denote underlying asset price and volatility at time t_j. Given the initial values of S_0 and V_0, we can simulate n underlying price dynamics to obtain S_T and the option price may be estimated using Equation (5).

Let $\vec{w}_0$ denote the parameters of original unobservable parameters of Heston model, e.g., the initial volatility V_0 and the correlation ρ, and let $\vec{w}_1, \vec{w}_2, \vec{w}_3, \vec{w}_4$ denote the weights of the neural networks NN_1, NN_2, NN_3, NN_4, respectively. Combine these parameters together and denote $\vec{w} = (\vec{w}_0, \vec{w}_1, \vec{w}_2, \vec{w}_3, \vec{w}_4)$. The calibration problem for the NSDE model shares the same form as Problem (2), except that $P(\vec{\theta}_i, \vec{w})$ is obtained through Monte Carlo simulation. Therefore, the calibration problem is now a simulation optimization problem where the objective function needs to be evaluated through simulation.

3 CALIBRATION OF THE NSDE MODEL

One with the knowledge of recurrent neural networks (RNN), see for instance Elman (1990), may find that the formulation and architecture presented in Equation (6) are quite similar to a RNN, which models the sequential output with hidden layers and given inputs. The architectures of Equations (6) and RNN have both similarities and differences. They both model sequential behaviors with neural networks as components and the previous stage's information as inputs. Apart from the similarities, there are also two main differences. First, the training of the RNNs tries to minimize the sum of differences between each stage's predicted outputs and the real outputs. So every stage's outputs should be taken into consideration. For the NSDE option pricing model, only the last stage's outputs are used in calibration. Second, RNN models contain a hidden layer at each stage whereas the NSDE model does not. The formulation of Equation (6) combines two forward sequential neural network models together. A comparison of the two models with three stages are illustrated in Figures 1 and 2. In Figure 1, x_t, s_t and o_t denote input, hidden layer and output at time t, respectively. In Figure 2, x_t, S_t, V_t denote related variables like the risk free rate and time to maturity, underlying asset price and volatility at time t, respectively.

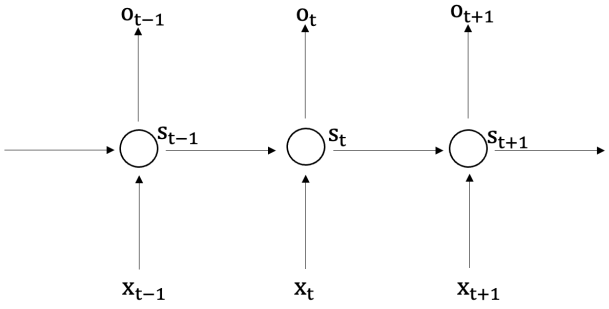


Figure 1: RNN model with three stages.

In neural network framework, weights are typically trained through the stochastic gradient descent (SGD) algorithm with the stochastic gradient computed using the back propagation (BP) algorithm. Hardt et al. (2016) showed that although there are no theoretical guarantees for the SGD algorithms in solving non-convex optimization problems, it is still a fast and stable algorithm for training neural networks. Because a RNN may be viewed as a sequence of neural networks, it can also be trained using the same approach (LeCun et al. 1998). Furthermore, since our NSDE model is similar to the RNN model, we can

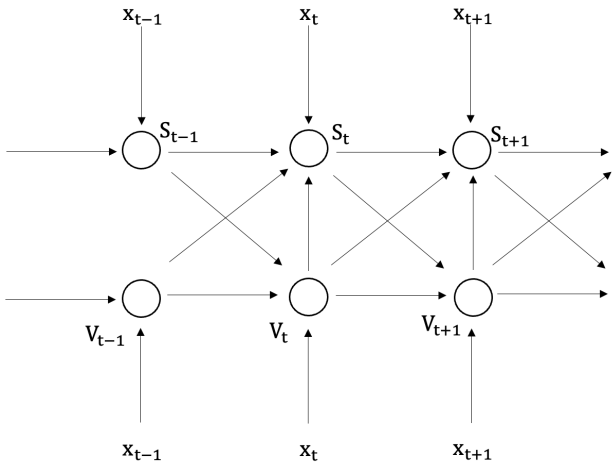


Figure 2: Euler's scheme of the NSDE model with three stages.

also adapt the approach to calibrate (or to train) our model. Then, the critical issue is how to use the BP algorithm to compute the gradient based on the simulated sample paths.

We use w_1 as an example to show how to apply the BP algorithm to derive the partial derivative with respect to w_1. Let L denote the loss function

$$L = \sum_{i=1}^{N} \left[P_i - P(\vec{\theta}_i, \vec{w}) \right]^2.$$

Then,

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial P(\vec{\theta}_i, \vec{w})} \frac{\partial P(\vec{\theta}_i, \vec{w})}{\partial w_1} = 2 \sum_{i=1}^{N} \left[P(\vec{\theta}_i, \vec{w}) - P_i \right] \frac{\partial P(\vec{\theta}_i, \vec{w})}{\partial w_1}. \tag{7}$$

Separate $\frac{\partial P(\vec{\theta}_i, \vec{w})}{\partial w_1}$ out, we get the following estimator according to Equation (5),

$$\frac{\partial P(\vec{\theta}_i, \vec{w})}{\partial w_1} = e^{-rT_i} \frac{1}{n} \sum_{k=1}^{n} \left[\frac{\partial S_{T_i}^k}{\partial w_1} \mathbf{I}_{\{S_T^k - K_i > 0\}} \right], \tag{8}$$

where $\mathbf{I}_{\{\cdot\}}$ denotes an indicator function. Denote $NN_1(S_{t_j}^k, V_{t_j}^k, r_f, t_j)$ and $NN_2(S_{t_j}^k, V_{t_j}^k, r_f, t_j)$ as $f_{t_j}^k$ and $g_{t_j}^k$ for short. Expanding $\frac{\partial S_{T_i}^k}{\partial w_1}$, we have

$$\frac{\partial S_{T_i}^k}{\partial w_1} = \sum_{j_1=0}^{m-1} S_{t_{j_1}}^k \frac{\partial f_{t_{j_1}}^k}{\partial w_1} \prod_{j_2=j_1+1}^{m-1} \left(1 + f_{t_{j_2}}^k \Delta t + g_{t_{j_2}}^k \sqrt{\Delta t} Z_{t_{j_2}}^{1k} + \frac{\partial f_{t_{j_2}}^k}{\partial S_{t_{j_2}}^k} S_{t_{j_2}}^k \right). \tag{9}$$

With Equations (7), (8) and (9), we have the estimator of the partial derivative with respect to w_1 using the BP algorithm. The partial derivatives with respect to w_2, w_3 and w_4 can be derived in a similar way.

The structure of Equation (9) helps mitigating vanishing gradients, a well known problem in neural network trainings. Take RNNs for example, we have the following equation for the partial derivative with respect to $\mathbf{u}$ in RNN,

$$\frac{\partial L_t}{\partial \mathbf{u}} = \sum_{k=0}^{t} \frac{\partial L_t}{\partial O_t} \frac{\partial O_t}{\partial S_t} \left(\prod_{j=k+1}^{t} \frac{\partial S_j}{\partial S_{j-1}} \right) \frac{\partial S_k}{\partial \mathbf{u}}, \tag{10}$$

where L_t denotes the loss function at time t. Notice that Equation (10) contains a product of many terms, i.e., $\prod_{j=k+1}^{t} \frac{\partial S_j}{\partial S_{j-1}}$. A small value of $\frac{\partial S_j}{\partial S_{j-1}}$ may cause the product close to zero, which is known as vanishing gradients (Li et al. 2018). Notice that in Equation (9), each term in the product has a constant 1 in it, so the product is unlikely to be close to 0. Therefore, this structure naturally mitigates the issue of vanishing gradients and makes the training more stable.

The above discussions focus on the calibration of the weights w_i of the neural networks. The NSDE model also has other unobservable parameters, such as V_0 and ρ. These parameters may be calibrated using similar approaches. Furthermore, as our model is based on Heston model, we can first calibrate the Heston model and use the calibrated values of these parameters as the initial values for the SGD algorithm. Once all unobservable parameters of the NSDE model are calibrated, option prices can be estimated using Monte Carlo methods.

Now we can summarize the entire NSDE algorithm for option pricing. Let $C_1, \ldots, C_N$ denote the historical option prices with strike prices $K_1, \ldots, K_N$ and expiration dates $T_1, \ldots, T_N$. We have the following algorithm.

The NSDE Algorithm for Option Pricing

Setup:

- Select appropriate neural network architectures for NN_1, NN_2, NN_3 and NN_4 with moderate numbers of layers.
- Set the time step Δt in Equation (6) with a proper length so that every T_i can be captured in $t_j, j = 1, \ldots, m$. Let T denote the end of the time intervals and m denote the number of time steps. Notice that $\Delta t = T/m$.

Initialization:

- Calibrate the initial values of the volatility V_0 and the correlation ρ using the closed-form pricing formula of Heston model.

Calibration:

1. Set the number of epochs D and the number of simulation sample paths n for each epoch.
2. Generate two $n \times m$ matrices $\mathbf{Z_1}$ and $\mathbf{Z_2}$ with all elements in them following the standard normal distribution, where the correlations of corresponding elements in two matrices are ρ.
3. With given S_0 and V_0, simulate n sample paths of the underlying asset S_t using $\mathbf{Z_1}$ and $\mathbf{Z_1}$ using Equation (6) and record them in a matrix $\mathbf{S}$. Take an element S_{ij} in $\mathbf{S}$ for example, the i-th row denote the i-th simulated sample path and j-th column denote the corresponding underlying asset price at time t_j of all n simulated sample paths. Calculate every predicted option price with the following equation
$$\hat{C}_i = \frac{1}{n} \sum_{k=1}^{n} e^{-r_t T_i} \left(S_{T_i}^k - K_i \right)^+,$$
where $S_{T_i}^k$ may be obtained from the corresponding column of the matrix $\mathbf{S}$ with the time equals to T_i.

4. Calculate the loss $L = \sum_{i=1}^{N} (\hat{C}_i - C_i)^2$. Calibrate the parameters of the model using the SGD algorithm with the gradient $\nabla L / \nabla \vec{w}$ computed using the BP algorithm. Update the parameters and go back to step 2 if the current number of epochs is less than D.

Option pricing:

- Given a set of new options of the underlying asset with different initial underlying assets price S_0, initial volatility V_0, strike prices and expiration dates. The option prices can be estimated using the new sample paths generated with these new inputs and the calibrated parameters using the same method as (2) and (3) in *Calibration.*

Remark 1. The number of layers and the width of every layer for NN_1, NN_2, NN_3 and NN_4 should not be too big. A deeper and wider neural network has a larger number of parameters, which need more data to calibrate. Because the training data sizes of option pricing problems are typically limited and the NSDE model is based on a well studied scientific model, small neural networks often work well in our problems. For more discussions on the structures of neural networks, please see Bergstra and Bengio (2012).

4 PRELIMINARY NUMERICAL RESULTS

In this section, to test the performance of the NSDE model and the NSDE algorithm, we conduct three numerical experiments. The first two experiments use option prices generated by the modified Heston model and the variance-gamma (VG) model, respectively. The third experiment uses the option prices on S&P 500 observed in the real financial market. To understand the performance of the NSDE model, we also compare it with some traditional option-pricing models and a direct neural-network model. We use the mean absolute error (MAE) to measure the prediction accuracy when comparing different models.

4.1 Modified Heston Model

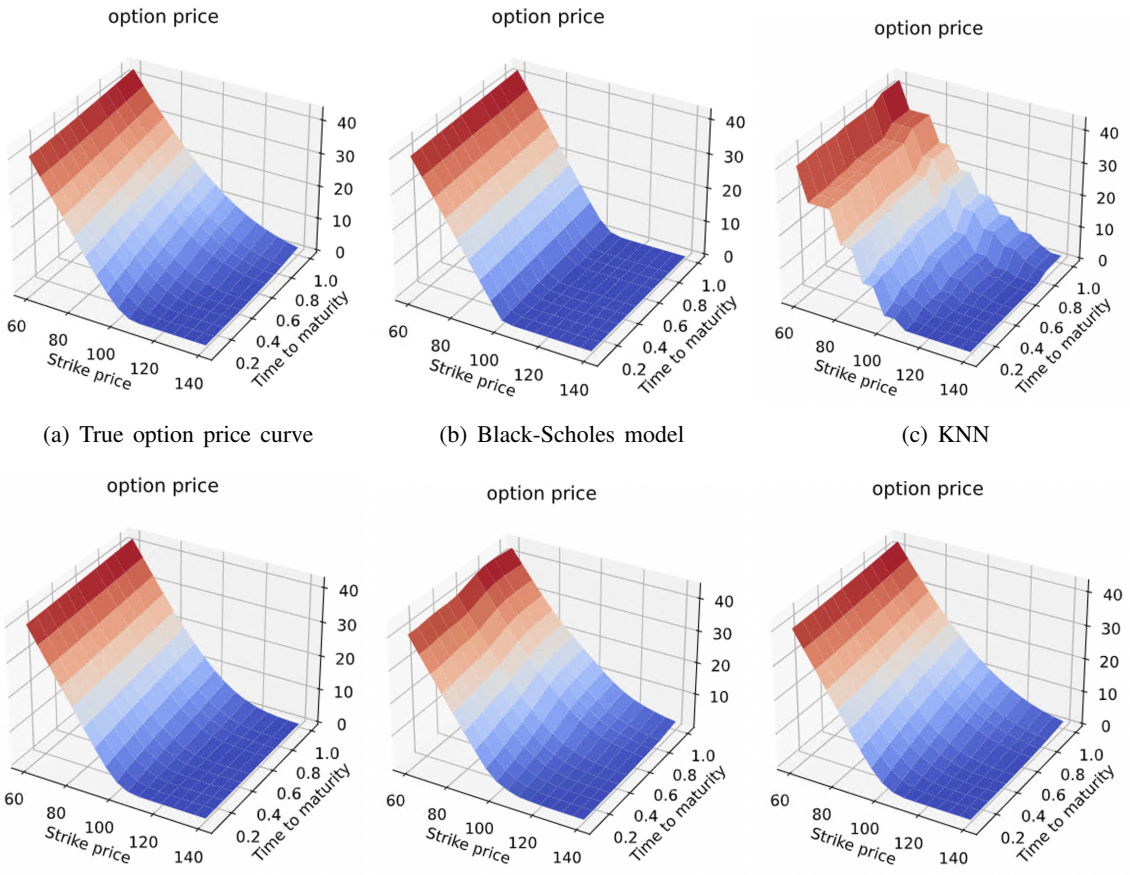
In the first experiment, the option prices are generated using a modified Heston model as follows:

$$\begin{aligned} dS_t &= \mu S_t dt + \sqrt{V_t} S_t dB_{1t} \\ dV_t &= \kappa(\theta - V_t)\, dt + \sigma dB_{2t}. \end{aligned} \tag{11}$$

Call option price data $P_1, \cdots, P_N$ are generated with a Monte Carlo simulation method. Notice that the model may be viewed as a special case of the NSDE model. For this numerical experiment, we set $r_f = 0.025$, $\kappa = 1.5$, $\sigma = 0.3$, $S_0 = 100$ and $V_0 = 0.04$. The training dataset is obtained by setting the expiration time $T = [1/12, 2/12, 3/12, 6/12, 1]$ whose unit is year and the strike price $K = [60, 70, 80, 90, 100, 110, 120, 130, 140]$. The testing dataset is obtained by setting the expiration time $T = [1/12, 2/12, 3/12, 4/12, 5/12, 6/12, 8/12, 9/12, 10/12, 1]$ and the strike price $K = [60, 65, 70, 75, 80, 85, 90, 95, 100, 105, 110, 115, 120, 125, 130, 135, 140]$. We compare our model with four baseline models: Black-Scholes model, Heston model, k-nearest neighbors (KNN) and the gated neural network (GNN) model of Yang et al. (2017). GNN is also a option pricing model that based on neural networks. The difference between our model and GNN is that GNN relied entirely on neural networks with a well-designed network structure for option pricing problems while our model used neural networks as a component of a well-studied option pricing model. Figure 3 shows the option price surfaces of these models. The MAE of these models are summarized in Table 1. From the figure and the table, We can see that the performance of the NSDE model is significantly better than other models for this example.

4.2 Variance-Gamma Model

In the second experiment, the option prices $P_1, \cdots, P_N$ are generated using a simulation method of the VG model of Fu (2007). Notice that the model is not a special case of the NSDE model. We set $\theta = -0.1436$,



(a) True option price curve (b) Black-Scholes model (c) KNN

(d) Heston model (e) Gated neural network (f) Neural SDE

Figure 3: The option price surfaces of different models for Experiment 1.

$\nu = 0.1686$, $\sigma = 0.1231$ and $S_0 = 100$. The training and testing sets are obtained by the same settings of T and K in Section 4.1. We also compare with the same five models of Section 4.1. Figure 4 shows the option price surfaces of these models. The MAE of these models are summarized in Table 2. From the figure and the table, We can see that, even though the true model is not a special case of the NSDE model, the performance of the NSDE model is still significantly better than other models for this example.

4.3 S&P500 ETF Index Options

In the third example, we use the S&P 500 ETF index option quotes observed in the real market as the dataset. We compare the NSDE model with Black-Scholes model and the GNN model. Figure 5 shows the errors of these models for all testing data points. The MAE of these models are summarized in Table 3. From Table 3, we notice that the MAEs of the GNN models are slightly better than those of the NSDE models. However, we also observe from Figure 5 that, at the outskirt of the input region, the performance

Table 1: The MAEs of different models for Experiment 1.

Model	BS	Heston	KNN	GNN	NSDE
Train MAE	1.45	1.46	1.73	0.53	0.33
Test MAE	1.63	1.35	1.95	0.62	0.43

Table 2: The MAEs of different models for Experiment 2.

Model	BS	Heston	KNN	GNN	NSDE
Train MAE	0.73	0.67	1.44	0.45	0.24
Test MAE	0.58	0.53	1.57	0.51	0.29

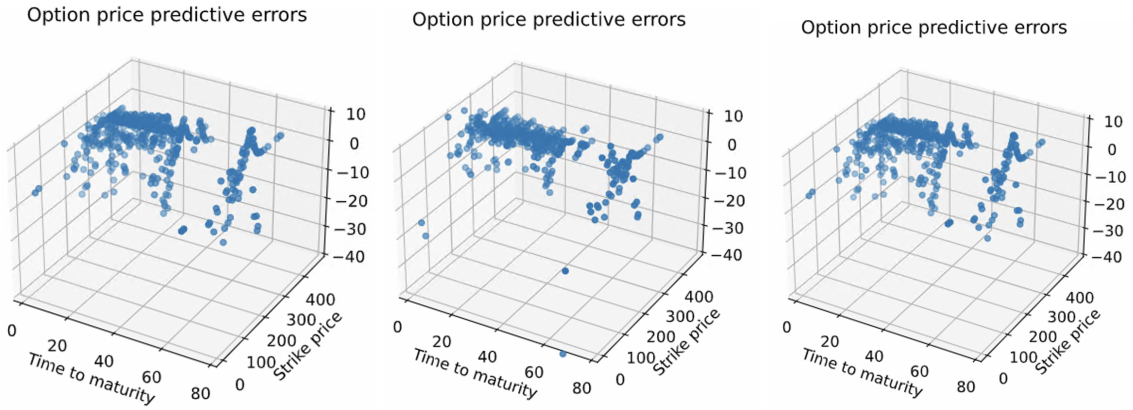
of the GNN model appears worse than that of the NSDE model. To further confirm this observation, we summarize the the performance of the NSDE and GNN models for different ranges of the strick price K in Table 4. We also find that the GNN model outperforms the NSDE model when K is in the middle, and under-performs when K is at the outskirt. This results demonstrate that, when there are sufficient amount of data, the GNN model that fits the option prices directly may work well. However, when the amount of data is limited, the SDE structure of the NSDE model helps significantly. Notice that the S&P 500 ETF index options are among the most popular options in the market. For many other options, the amount of data is typically significantly smaller. In these situations, the NSDE model may perform significantly better than other models.

Table 3: The MAEs of different models for Experiment 3.

Model	BS	GNN	NSDE
Train MAE	4.6	3.0	3.5
Test MAE	5.7	3.4	3.7

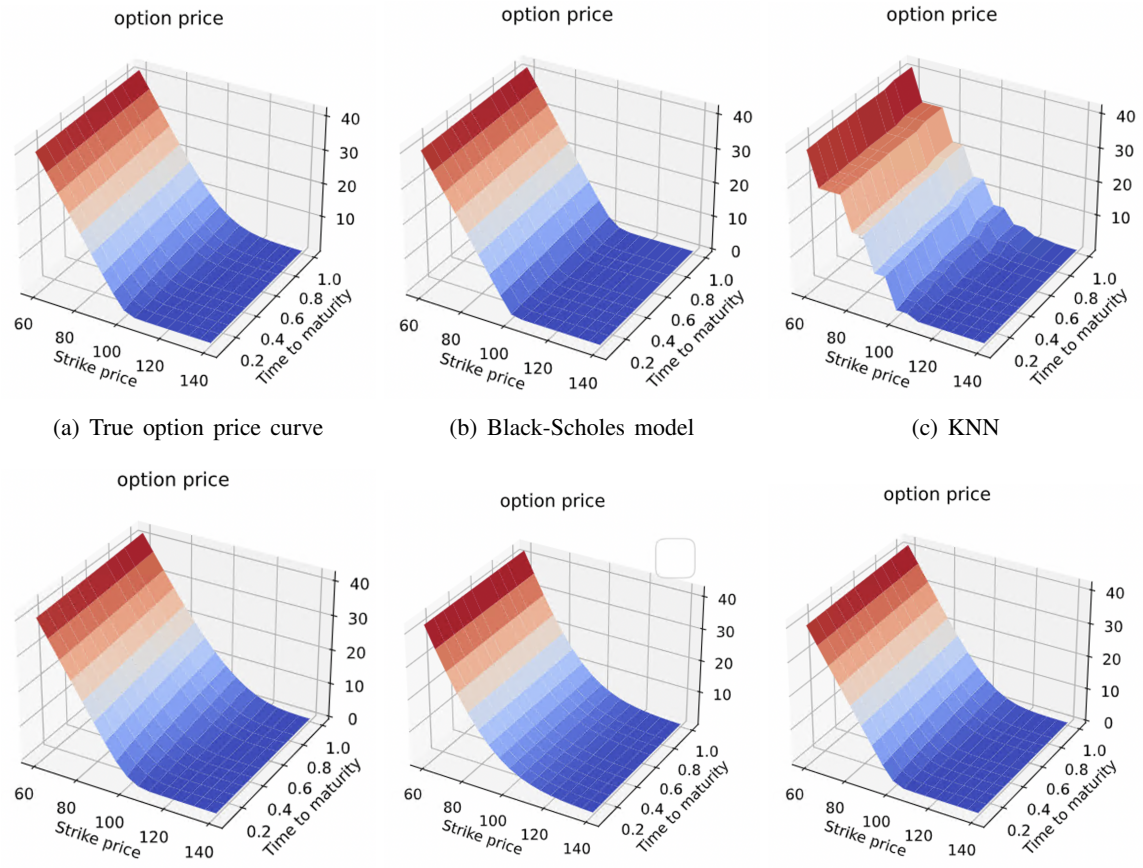
Table 4: The MAEs of NSDE and GNN for Experiment 3 with different ranges of K.

Quantile of K	<10%	10%-90%	>90%
NSDE MAE	6.5	3.8	1.6
GNN MAE	9.6	3.1	1.9



(a) Black-Scholes (b) Gated neural network (c) Neural SDE

Figure 5: Errors in Experiment 3.



(a) True option price curve (b) Black-Scholes model (c) KNN

(d) Heston model (e) Gated neural network (f) Neural SDE

Figure 4: The option price surfaces of different models for Experiment 2.

5 CONCLUSIONS

In this paper we develop a new NSDE model for option pricing by integrating neural networks as components to the famous Heston model. On one hand, this model is less rigid than the original Heston model, and it is capable of capturing more market characteristics. On the other hand, the model builds on the stochastic volatility structure that has been well studied and widely accepted in the literature, and it requires smaller data sets than direct neural network models. A preliminary numerical study supports our insights but also calls for more in-depth studies.

REFERENCES

Bakshi, G., C. Cao, and Z. Chen. 1997. "Empirical performance of alternative option pricing models". *The Journal of Finance* 52(5):2003–2049.

Bates, D. S. 2003. "Empirical option pricing: A retrospection". *Journal of Econometrics* 116(1-2):387–404.

Bergstra, J., and Y. Bengio. 2012. "Random search for hyper-parameter optimization". *Journal of machine learning research* 13(2):281–305.

Black, F., and M. Scholes. 1973. "Theory of rational option pricing". *The Bell Journal of Economics and Management Science* 4(1):141–183.

Chen, R. T., Y. Rubanova, J. Bettencour, and D. Duvenaud. 2018. "Neural ordinary differential equations". In *Advances in neural information processing systems*. December 2th-8th, Montreal, Canada, 6571-6583.

Cont, R., and P. Tankov. 2004. *Financial modelling with jump processes*. 2nd ed. Boca Raton, Florida: CRC Press.

Elman, J. L. 1990. "Finding structure in time". *Cognitive Science* 14(2):179–211.

Fu, M. C. 2007. *Variance-gamma and Monte Carlo*. 1st ed. Boston, Massachusetts: Birkhäuser Boston.

Hardt, M., B. Recht, and Y. Singer. 2016. "Train faster, generalize better: Stability of stochastic gradient descent". In *International Conference on Machine Learning*. June 19th-24th, New York City, USA, 1225-1234.

Heston, S. L. 1993. "A closed-form solution for options with stochastic volatility with applications to bond and currency options". *The Review of Financial Studies* 6(2):327–343.

Heston, S. L., and S. Nandi. 2000. "A closed-form GARCH option valuation model". *The Review of Financial Studies* 13(3):585–625.

Hornik, K., M. Stinchcombe, and H. White. 1989. "Multilayer feedforward networks are universal approximators". *Neural Networks* 2(5):359–366.

Kou, S. G. 2002. "A jump-diffusion model for option pricing". *Management Science* 48(8):1086–1101.

LeCun, Y., L. Bottou, Y. Benjio, and P. Haffner. 1998. "Gradient-based learning applied to document recognition". *Proceedings of the Institute of Electrical and Electronics Engineers, Inc.* 86(11):2278–2324.

Li, S., W. Li, C. Cook, C. Zhu, and Y. Gao. 2018. "Building a longer and deeper rnn". In *Proceedings of the Institute of Electrical and Electronics Engineers, Inc. conference on computer vision and pattern recognition*. June 19th-21th, Salt Lake City, USA, 5457-5466.

Madan, D. B., P. P. Carr, and E. C. Chang. 1998. "The variance gamma process and option pricing". *Review of Finance* 2(1):79–105.

Merton, R. C. 1976. "Option pricing when underlying stock returns are discontinuous". *Journal of Financial Economics* 3(1-2):125–144.

Metron, R. C. 1973. "The Pricing of Options and Corporate Liabilities". *The Journal of Political Economy* 81(3):637–654.

Raissi, M., P. Perdikaris, and G. E. Karniadakis. 2019. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". *Journal of Computational Physics* 378:686–707.

Strikwerda, J. C. 1989. *Finite-Difference Schemes and Partial Differential Equations*. 1st ed. Pacific Grove, California: Wadsworth and Brooks.

Yang, Y., Y. Zheng, and T. Hospedales. 2017. "Gated neural networks for option pricing: Rationality by design". In *Association for the Advancement of Artificial Intelligence*. February 4th-9th, San Francisco, USA, 52-58.

AUTHOR BIOGRAPHIES

SHOUDAO WANG is a master student in the School of Data Science at Fudan University. His research interests include stochastic modeling, simulation optimization and machine learning methods with applications in financial engineering. His email address is 19210980104@fudan.edu.cn.

L. JEFF HONG is Fudan Distinguished Professor and Hongyi Chair Professor with joint appointment at School of Management and School of Data Science at Fudan University in Shanghai, China. His research interests include stochastic simulation, stochastic optimization, risk management and supply chain management. He is currently the simulation area editor of *Operations Research*, an associate editor of *Management Science* and the President of INFORMS Simulation Society. His email address is hong_liu@fudan.edu.cn.