Option pricing and volatility surface based on physics-informed neural network

Hyeong-Ohk Bae
Dept of Financial engineering
Ajou University
Republic of Korea
hobae@ajou.ac.kr

Seunggu Kang
Dept of Financial engineering
Ajou University
Republic of Korea
crator777@ajou.ac.kr

Muhyun Lee
Samsung Securities
11 Seocho-daero 74-gil, Seocho-gu, Seoul
moo.lee@samsung.com

April 3, 2023

Abstract

We solve the Black-Scholes equation and constant elasticity of variance model for European options under the local volatility model by an artificial neural network, and calculate the price and the Greeks simultaneously.

We also apply physics-informed neural network to the multidimensional Black-scholes equation and calculate the price of a step-down ELS which has two underlying assets. Then we compare the prices and Greeks by PINN and by OSM.

Keywords: Option pricing, Local volatility, Neural Network, Black-Scholes equation (BSE), Dupire's equation, Physics-informed neural network (PINN)

1 Introduction

The Black-Scholes partial differential equation (BSE) [3] is the most widely used option pricing model. The assumption that the price of the underlying asset follows a log-normal process with a constant volatility, is useful for practitioners because there is closed-form solutions for the European options.

However, the constant volatility assumption is not realistic in the real world. Dupire's local volatility model is most useful in practice and it is important to calculate rices and Greeks under the local volatility.

In practice, estimating the volatility surface is important for pricing exotic options or hedging derivatives. Since different implied volatilities are observed depending on maturity and strike price in the market, after works by [8, 9] many studies have been conducted to estimate the volatility surface via the local volatility model ([4, 5, 11, 18, 21]). In [26], by using the parameters obtained by calibration and the implied volatility formula under SABR model, the authors provide several quantitative properties of volatility by drawing volatility surface.

A local volatility model does not have a closed form solution. As numerical methods, the Monte-Carlo method and the finite difference method (FDM) are most popular for option pricing. However, they have some disadvantages. Typically, price and Greeks cannot be obtained at the same time. In [16], the mesh-free point collocation method is used to calculate option price and their Greeks simultaneously. Neural networks can be used in a sense similar to the mesh-free methods, which compute price and Greeks simultaneously. It also solves parametric partial differential equations (PDEs), and approximates even the derivatives with respect to the parameter (see Section 2.3 and [20]).

Research on solving PDE using artificial neural networks has been in progress for several decades [19, 17, 22, 27], and many studies [24, 23, 13, 2] have shown the deep learning as an interesting tool. In [23], a deep learning framework for solving nonlinear PDEs is proposed, called the physics informed neural network (PINN). Owing to the fact that a physical modeling is given in advance instead of providing data, it is different from usual machine learning algorithms treated as black box tools. In a usual deep learning, the automatic differentiation technique is used to only the back propagation process. Differently from a usual deep learning, PINN uses automatic differentiation to calculate derivatives included in a PDE where a physical information is described.

In this article, using PINN as a neural network approach, we calculate solutions of single and multi-dimensional BSEs, Constant Elasticity of Variance (CEV), Dupire's PDE, related volatilities and prices for European options. We also calculate price and Greeks of exotic options. Then we compare prices and Greeks calculated by PINN and by Operator splitting method (OSM).

As an exotic option and one of the flag ship products of the investment bank in Korea, Equity Linked Security (ELS) is a security whose returns

on investment are tied to the returns of individual stocks or equity indices. There are different types of ELS depending on the payoff structures. Among them, step-down ELS is a representative product.

The contribution of this article is fourfold.

1. We adopt a neural network algorithm to solve a family of extended BSEs, for example, CEV and Dupire's local volatility model. Particularly, after training the network in the offline phase, the network estimates solutions for different parameter values in milliseconds.
2. We construct a local volatility surface and then, as an application, we show the practical usage of the method.
3. To evaluate the performance of the method, we study the errors in the option price and in Dupire's equation for local volatility models.
4. We apply PINN to the multi-dimensional BSE and calculate the price and Greeks of a two-asset step-down ELS. Then we compare the prices and Greeks by PINN and by OSM.

This article is based on master dessertations [15, 20].

2 Solving Black-Scholes PDE with Local Volatility via Physics-Informed Neural Network

2.1 Local Volatility model

A well-known limitation of the Black-Scholes model is the assumption of a constant volatility. However, the implied volatilities in the market depend on the strike price and time to maturity. Implied volatility tends to be higher in deep Out of The Money(OTM) options and deep In The Money(ITM) options than At The Money(ATM) options. It is explained by a phenomenon called volatility smile (or skew) (refer to [9, 8]). In the local volatility model, in a risk-neutral world, the volatility is assumed as a deterministic function of time and stock price in the spreading period of the stock price process as follows:

$$dS_t = rS_t dt + \sigma(S_t, t)S_t dW_t, \tag{2.1}$$

where S_t is a stock price at time t, r is a constant risk-free rate, and dW_t is the standard Wiener process in the risk-neutral world. The local volatility $\sigma(S_t, t)$ is a function of the underlying asset price S_t and the time t. In [6] denoted by $V(S_t, t)$ a price function of a derivative with underlying asset S_t.

The extended BSE under the no arbitrage condition is as follows:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2(S_t, t)S_t^2\frac{\partial^2 V}{\partial S_t^2} + rS_t\frac{\partial V}{\partial S_t} - rV = 0.$$

Assuming that the price of the underlying asset follows (2.1), the price of a European call option is given by $c(S_0, K, T) = e^{-rT}\mathbb{E}[(S_T - K)^+]$, where K is the strike price of the option, T is the maturity and $\mathbb{E}$ is the expectation. In [9], a volatility function is obtained by solving the following PDE:

$$\sigma^2(K, T) = 2\frac{\frac{\partial c}{\partial T} + rK\frac{\partial c}{\partial K}}{K^2\frac{\partial^2 c}{\partial K^2}} = 2\frac{\frac{\partial p}{\partial T} + rK\frac{\partial p}{\partial K}}{K^2\frac{\partial^2 p}{\partial K^2}}, \tag{2.2}$$

where p is the correspoding put option. If the local volatility is constant with respect to price and time, then (2.1) is reduced to the geometric Brownian motion (or called the Black-Scholes model in practice). A model leading to the skew of implied volatility is the CEV model [6, 7].

2.1.1 Geometric Brownian Motion

The geometric Brownian motion (GBM) follows log-normal, and has the advantage that the existence of closed-form solutions for various options is well known. It can be seen as the simplest model of the local volatility model as flat surface. The stochastic differential equation for GBM is the following:

$$dS_t = rS_t dt + \sigma S_t dW_t, \tag{2.3}$$

where σ is a constant. The solutions for European options are known as the Black-Scholes formula:

$$\begin{aligned} c(S_t, t, K, r, \sigma) &= S_t N(d_1) - Ke^{-r(T-t)}N(d_2), \\ p(S_t, t, K, r, \sigma) &= Ke^{-r(T-t)}N(-d_1) - S_t N(-d_2), \end{aligned}$$

where $d_1 := \frac{\log(S_t/K) + (r + \sigma^2/2)(T-t)}{\sigma\sqrt{T-t}}$, $d_2 := d_1 - \sigma\sqrt{T-t}$ and $N(x)$ is the cumulative normal distribution. The advantage of a model with a closed solution is that Greeks can be found analytically. Greeks' analytic formula are used for model evaluation. For Greeks, we refer to [18] which contains many formula.

2.1.2 Constant Elasticity of Variance model

The constant elasticity of variance (CEV) model [7] with the local volatility $\sigma(S_t, t) = \sigma S_t^{(\beta/2)-1}$ with $\beta \in \mathbb{R}$, explains the negative skew on the underlying asset's price, and there are closed-form solutions for European vanilla options. CEV model is the same as that of GBM when $\beta = 2$, and the square-root diffusion model in [6] when $\beta = 1$. In [10] the closed-form solution is provided for European call option of $\beta > 2$. Equations for CEV model are as follows:

$$\begin{aligned} dS_t &= rS_t dt + \sigma S_t^{\beta/2} dW_t, \\ \frac{\partial V}{\partial t} &+ \frac{1}{2}\sigma^2 S_t^{\beta} \frac{\partial^2 V}{\partial S_t^2} + rS_t \frac{\partial V}{\partial S_t} - rV = 0, \end{aligned} \tag{2.4}$$

where σ is a constant. The closed form solution for European put option $p(S_t, t, K, r, \sigma)$ is

$$\begin{aligned} &p(S_t, t, K, r, \sigma) \\ &= \begin{cases} Ke^{-r(T-t)} Q(2x; \frac{2}{2-\beta}, 2y) - S_t\big[1 - Q(2y; 2 + \frac{2}{2-\beta}, 2x)\big], & \text{for } \beta < 2, \\ Ke^{-r(T-t)} Q(2y; 2 + \frac{2}{\beta-2}, 2x) - S_t\big[1 - Q(2x; \frac{2}{\beta-2}, 2y)\big], & \text{for } \beta > 2, \end{cases} \end{aligned} \tag{2.5}$$

where $Q(w; v, \lambda)$ is the complementary distribution function of a non-central chi-square law, v is the degree of freedom, λ is a non-centrality parameter, $x = S_t^{2-\beta} e^{r(2-\beta)(T-t)} d$, $y = K^{2-\beta} d$, $d = 2r/\big\{\delta^2(2-\beta)[e^{r(2-\beta)(T-t)} - 1]\big\}$, and $\delta^2 = \sigma_0^2 S_0^{2-\beta}$. These values are given in [17].

2.2 Neural Network for PDE solver

Artificial neural networks are essentially functions that map from input variables to output values. The layers in the neural network consist of composite functions of affine mapping (weighted sum) and nonlinear mapping (activation function) functions. The universal approximation theorem has been demonstrated in [12] that a neural network with one hidden layer can approximate an arbitrary continuous function. This shows the neural network is a kind of function approximation, and as a result, it can be used as a method to solve PDE. Recently, neural network techniques to solve PDEs has been developed actively. The automatic differentiation technique, which contributes greatly to the efficient learning of network weights and biases, provides partial derivatives of the neural network analytically. Therefore, this plays an important role in constructing a PDE for an objective function.

2.2.1 Neural Network for Local Volatility Black-Scholes Equation

Consider parametric BSE for $V(\tau, s, k)$ under the local volatility model:

$$\begin{aligned}
&\frac{\partial V}{\partial \tau} - \frac{1}{2}\sigma^2(\tau, s)s^2\frac{\partial^2 V}{\partial s^2} - rs\frac{\partial V}{\partial s} + rV = 0, \\
&\qquad\qquad\qquad\qquad \text{for } (\tau, s, k) \in [0, T] \times \Omega_S \times \Omega_K, \\
&V(0, s, k) = V_0(s, k), \qquad (s, k) \in \Omega_S \times \Omega_K, \\
&V(\tau, s, k) = g(\tau, s, k), \qquad (\tau, s, k) \in [0, T] \times \partial\Omega_S \times \Omega_K,
\end{aligned}$$

where $\tau = T - t$ is the time to maturity, $\Omega_S, \Omega_K \subset [0, \infty)$ are closed sets of stock price S and strike price K, respectively, $\partial\Omega_S$ is the boundary of Ω_S, V_0, g are given initial and boundary conditions.

We denote by $u^\theta(\tau, s, k) \in \mathbb{R}^1$ a functional form of artificial neural network for option price with respect to τ, stock price s, and strike price k. Artificial neural network $u^\theta(\tau, s, k)$ approximates $V(\tau, s, k)$, where θ is the neural network's parameter.

Construct the objective function for BSE under the local volatility model as a summatin of mean square error losses for the equation, initial and boundary conditions:

$$\begin{aligned}
&\mathcal{L}_{eq}(u^\theta) := \left\| \frac{\partial u^\theta}{\partial \tau} - \frac{1}{2}\sigma^2(\tau, s)s^2\frac{\partial^2 u^\theta}{\partial s^2} - rs\frac{\partial u^\theta}{\partial s} + ru^\theta \right\|^2, \\
&\mathcal{L}_{ic}(u^\theta) := \|u^\theta(0, s, k) - V_0(s, k)\|^2, \\
&\mathcal{L}_{bc}(u^\theta) := \|u^\theta(\tau, s, k) - g(\tau, s, k)\|^2, \\
&\mathcal{L}_{total}(u^\theta) := \mathcal{L}_{eq}(u^\theta) + \mathcal{L}_{ic}(u^\theta) + \mathcal{L}_{bc}(u^\theta),
\end{aligned} \tag{2.6}$$

where $\mathcal{L}_{total}(u^\theta)$ is the objective function. The notation $\|\cdot\|$ means the mean square error with respect to the discretizations of τ and/or s and k. If $\mathcal{L}_{total}(u^\theta) = 0$, then $u^\theta(\tau, s, k)$ is a solution of BSE.

In [12, 24], define by $\mathbb{C}^n$ the class of neural networks with a single hidden layer and n hidden units. Let u^n be a neural network with n hidden units which minimizes $\mathcal{L}_{total}(u^n)$. It is proved that, under certain conditions,

$$\begin{aligned}
&\text{there exists } u^n \in \mathbb{C}^n \text{such that } \mathcal{L}_{total}(u^n) \to 0, \text{ as } n \to \infty, \text{ and} \\
&u^n \to V \text{ as } n \to \infty.
\end{aligned} \tag{2.7}$$

The goal of training neural network is to find a set of parameter θ such that the function $u^\theta(\tau, s, k)$ minimizes the total loss function $\mathcal{L}_{total}(u^\theta)$. If the total error is small, then neural network approximate the solution of PDE. The algorithm is following:

Algorithm 1 Algorithm of PINN for local volatility BSE

Input: Vector of random spatial points (τ_i, s_i, k_i).
Output: Vector of solutions of parametric BSE $u^\theta(\tau_i, s_i, k_i)$.

Initialize epoch as 0.
while $\left(\mathcal{L}_{total}(u^\theta) > 1e-7\right)$ or epoch < 1000 **do**
  Initialize step as 0.
  Uniformly generate random points
  $(\tau_i^{(1)}, s_i^{(1)}, k_i^{(1)})$ from $[\theta, T] \times \Omega_S \times \Omega_K$
  $(s_i^{(2)}, k_i^{(2)})$ from $\Omega_S \times \Omega_K$
  $(\tau_i^{(3)}, s_i^{(3)}, k_i^{(3)})$ from $[0, T] \times \partial\Omega_S \times \Omega_K$
  while $\left(\mathcal{L}_{total}(u^\theta) > 1e-7\right)$ or step < 5000 **do**
    Calculate the total loss at the random sampled points where $\mathcal{L}_{total}$
    $= \mathcal{L}(u^\theta(\tau_i^{(1)}, s_i^{(1)}, k_i^{(1)})) + \mathcal{L}_{ic}(u^\theta(s_i^{(2)}, k_i^{(2)})) + \mathcal{L}_{bc}(u^\theta(\tau_i^{(3)}, s_i^{(3)}, k_i^{(3)}))$.
    Take a gradient descent step with Adam optimizer for θ.
    Add 1 to step.
  end while
  Add 1 to epoch.
end while

2.2.2 Transfer learning

In off-line phase, as the number of the parameters in neural networks increases, so is the accuracy of the approximation. However, computing time also increases. To overcome this, we use the transfer learning used in machine learning. With the transfer learning scheme, we first train a neural network in a local volatility model, and then using this pre-trained neural network, we train the neural network for other local volatility model.

We show that during the numerical simulation, the second training process converges faster than the first training. In the simulation we find in Tables 1 and 2, the computation and convergence rates are more efficient than those of random initialization network when adopting transfer learning. Table 1 shows the speed and error of the training process for the neural network in which the parameters are random initialized without the transfer learning scheme. Table 2 shows the speed and error of the training process with a pre-trained neural network of which the parameters had been fitted to a different local volatility model. These two models train the same volatility surface model. The tables show that the transfer learning is more efficient. As a result, transfer learning is efficient in both computational speed and convergence.

Step	Time (seconds)	Training error
1000	74.01	0.010114
2000	135.25	0.002174
4000	258.20	0.002027
8000	521.36	0.000427
16000	1032.19	8.85669E-05
32000	1984.73	5.97776E-06

Table 1: Random initialized neural network training

Step	Time (seconds)	Training error
1000	47.61	3.15432E-05
2000	82.07	1.05102E-05
4000	150.59	4.11163E-06
8000	299.56	2.51589E-06
16000	610.32	9.96727E-07
32000	1068.40	6.59060E-07

Table 2: Pre-trained neural network training

2.3 Numerical results

We implement the neural networks of local volatility models (2.1), (2.3) and (2.4) in TensorFlow2 on a GeForce RTX 3070. We describe our result in two ways. For local volatility models in which a closed form solution of European put option exists, we evaluate price and greeks(Delta, Gamma, Theta) with the neural networks. In case that there is no a closed form solution of European put option, we evaluate Dupire's equation (2.2) with neural networks

$$\sigma_{NN}^2 = 2\big(\frac{\partial u^\theta}{\partial T} + rK\frac{\partial u^\theta}{\partial K}\big)/\big(K^2\frac{\partial^2 u^\theta}{\partial K^2}\big). \tag{2.8}$$

If the neural network $u^\theta(\tau, s, k)$ is well trained with the object function (2.6) for European put options, then (2.8) should be equal to the local volatility function. Instead of the price of the underlying asset, moneyness is used and the maturity is fixed at one year $T = 1$. Risk-free interest rate is $r = 0.01$, volatility is $\sigma = 0.3$, strike price fixed $k = 1$, and for CEV model we consider $\beta = 1, 3$.

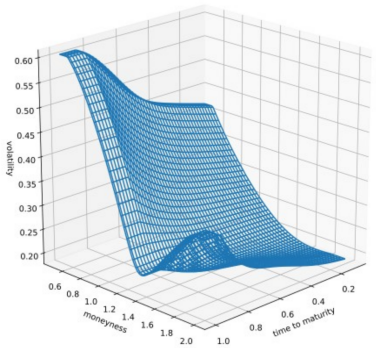


Figure 1: Local volatility surface obtained from EUROSTOXX50 option data of January 15, 2016.

In this subsection, for volatility surface model, we use the market European option data consisting of several maturities and strike prices of which underlying asset is EUROSTOXX50 as of January 15, 2016. We adopt [26] to derive

volatility surface from market data (see Figure 1). Based on the volatility surface in Figure 1 as a local volatility $\sigma(S_t, t)$ in (2.1), we calculate all prices and Greeks.

According to (2.7), the higher the number of hidden units, the better the approximation. We have adopted the number of hidden units 20,000. We have used the softplus function as an activation function.

2.3.1 GBM

For GBM, we fix $\sigma = 0.3$ as a plane volatility surface. Figures 2 and 3 show price, delta, gamma and theta of European put option with GBM. Figure 2 is obtained from the solution formula, and Figure 3 is calculated by PINN. In Figure 4, we evaluate L^2 errors of price, delta, gamma and theta. Table 3 shows L^2 errors of price and Greeks.

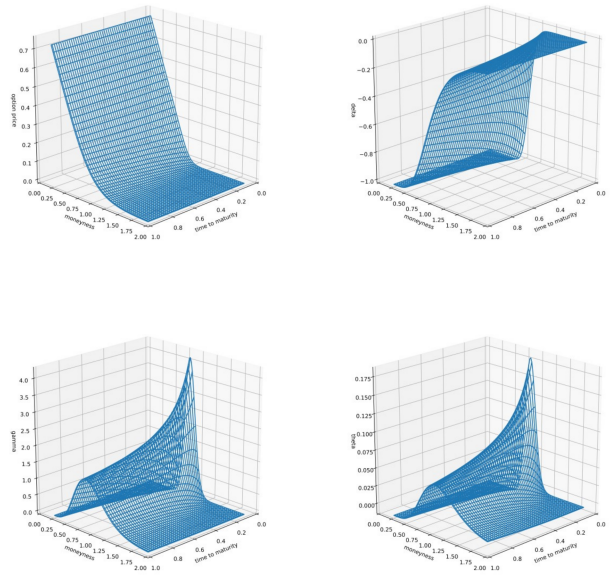


Figure 2: European put option price and Greeks with GBM: price(left top), delta(right top), gamma(left bottom), theta(right bottom)

Figure 5 shows the constant volatility and volatility obtained by solving (2.8) of GBM via PINN.

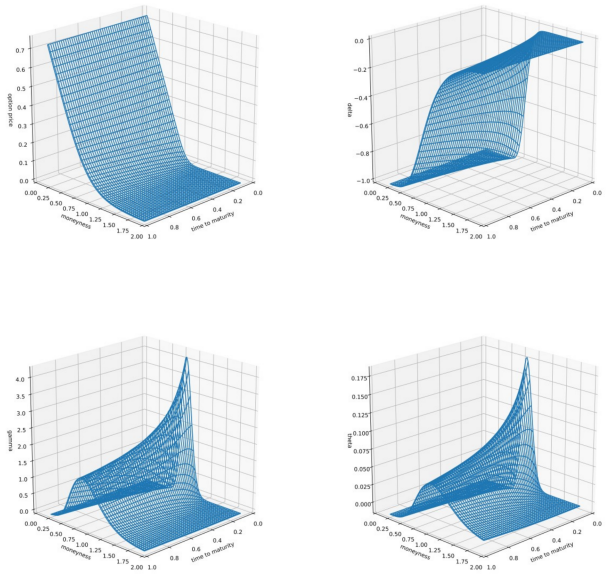
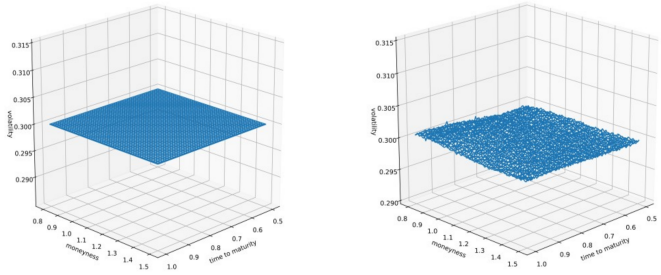


Figure 3: PINN approximation of European put option price and Greeks with GBM: price(left top), delta(right top), gamma(left bottom), theta(right bottom)



(a) Volatility surface of GBM (b) Dupire's equation with PINN with $\sigma = 0.3$

Figure 5: Constant volatility $\sigma = 3$ (left) and Volatility obtained by (2.8).

	Price	Delta	Gamma	Theta
Maximum L^2 error	0.00048	0.00201	0.02858	0.00103
Minimum L^2 error	4e-09	6e-14	2e-12	4e-13

Table 3: GBM Neural network's L^2 errors of price and Greeks

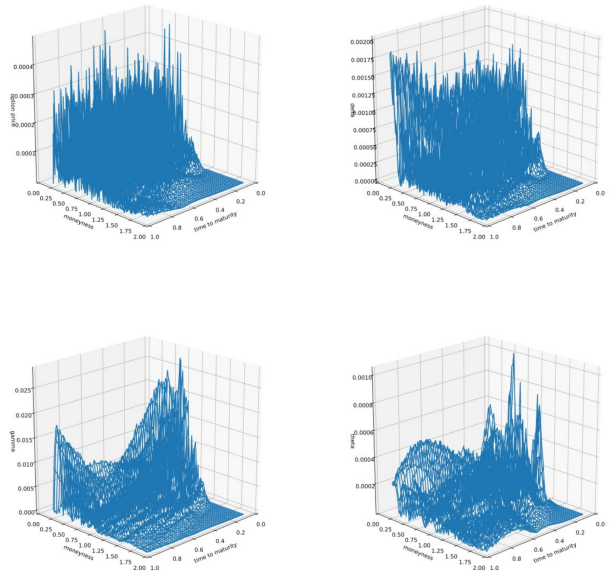


Figure 4: L^2 errors of GBM via PINN: price(left top), delta(right top), gamma(left bottom), theta(right bottom)

2.3.2 CEV model

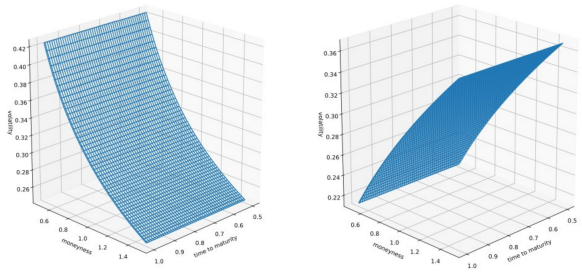
One of the extention of BSE is CEV model. We provide the results of whether the neural network (PINN) approximates the solution and Dupire's equation well for CEV model. From our results, we show that the neural network approximates the parametric PDE as well as its derivatives simultaneously.

We implement CEV for $\beta = 1$ and $\beta = 3$. With fixed $\sigma = 0.3$, (2.5) is used to evaluate for $\beta = 1, 3$. Volatility surfaces of CEV model with $\beta = 1, 3$ are shown in Figure 6. Prices, deltas, gammas, thetas of European put option with CEV obtained from the solution formula are privided in Figure 7 for $\beta = 1$, and in Figure 8 for $\beta = 3$. Table 4 provides L^2 errors of prices and Greeks obtained by PINN.

CEV ($\beta = 1$)	Price	Delta	Gamma	Theta
Maximum L^2 error	0.00048	0.00201	0.02858	0.00103
Minimum L^2 error	4e-09	6e-14	2e-12	4e-13
CEV ($\beta = 3$)	Price	Delta	Gamma	Theta
Maximum L^2 error	0.00055	0.00230	0.03415	0.00171
Minimum L^2 error	1e-09	4e-10	7e-09	3e-09

Table 4: CEV ($\beta = 1, 3$) Neural network's L^2 errors of price and Greeks

In Figures 9($\beta = 1$) and 10($\beta = 3$), prices, deltas, gammas and thetas of European put option with CEV model approximated by PINN are shown. In Figures 11($\beta = 1$) and 12($\beta = 3$), we evaluated L^2 error of price(left top), delta(right top), gamma(left bottom) and theta(right bottom) for CEV model. Table 4 and 5 shows error of price and Greeks. Figure 13 shows (2.8) of both CEV models. Dupire's equations of the CEV neural networks have a similar shape to the local volatility functions of CEV, indicating that the approximation of the parametric PDE is well done.



(a) $\beta = 1$ (b) $\beta = 3$

Figure 6: Volatility surfaces of CEV model with $\beta = 1$ and $\beta = 3$.

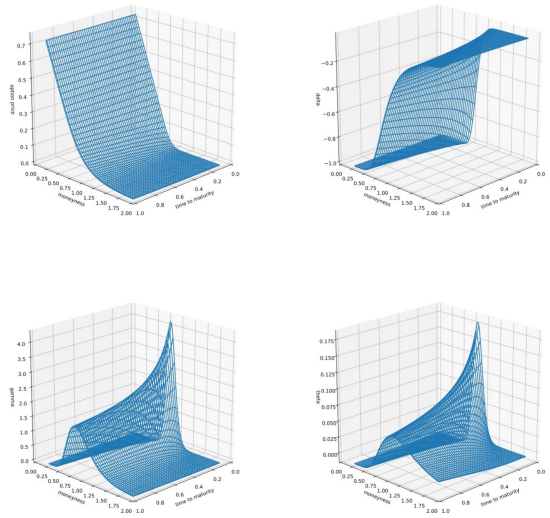


Figure 7: European put option price and Greeks with CEV ($\beta = 1$): price(left top), delta(right top), gamma(left bottom), theta(right bottom)

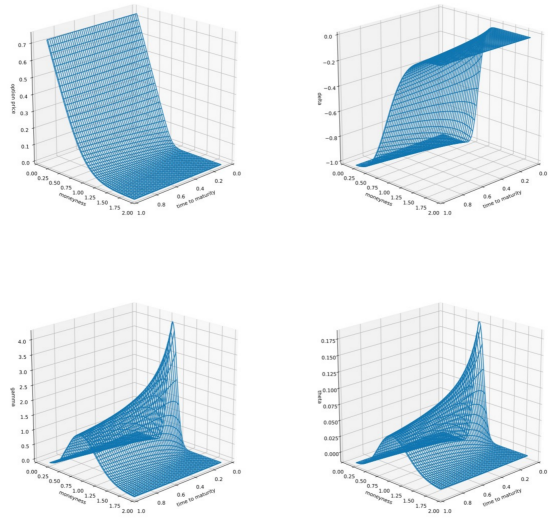


Figure 9: CEV ($\beta = 1$) PINN approximation of European put option price and Greeks price(left top), delta(right top), gamma(left bottom), theta(right bottom)

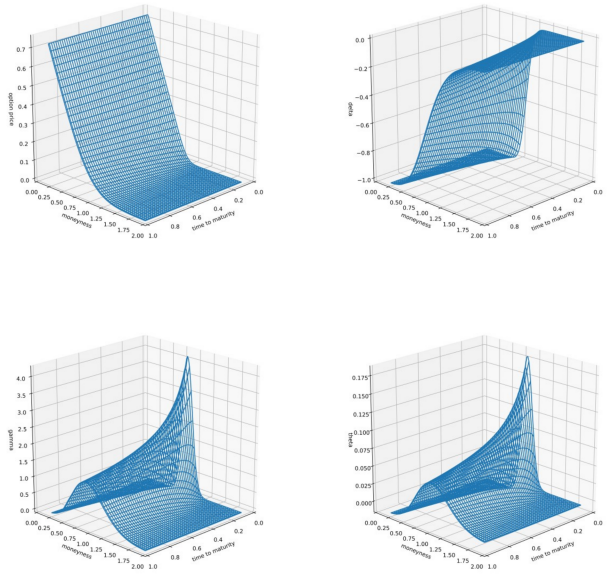


Figure 8: European put option price and Greeks with CEV ($\beta = 3$): price(left top), delta(right top), gamma(left bottom), theta(right bottom)

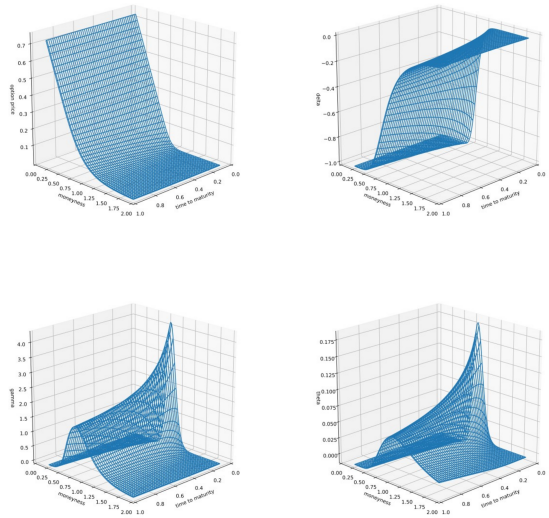


Figure 10: CEV ($\beta = 3$) PINN approximation of European put option: price(left top), delta(right top), gamma(left bottom), theta(right bottom)

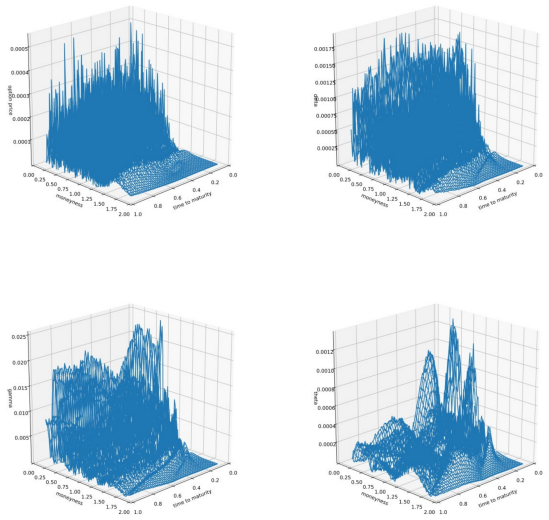


Figure 11: L^2 errors of CEV ($\beta = 1$) of PINN: price(left top), delta(right top), gamma(left bottom), theta(right bottom)

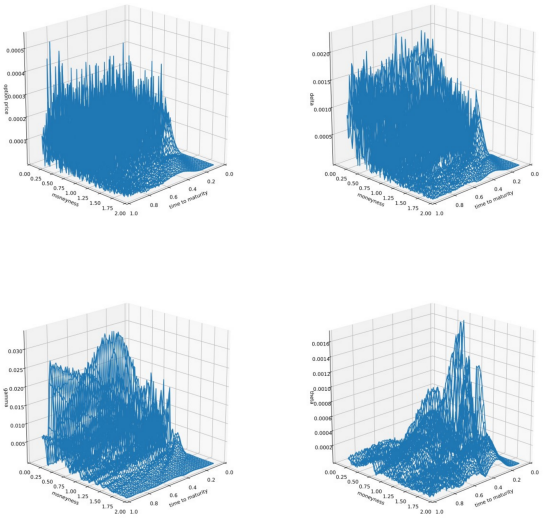
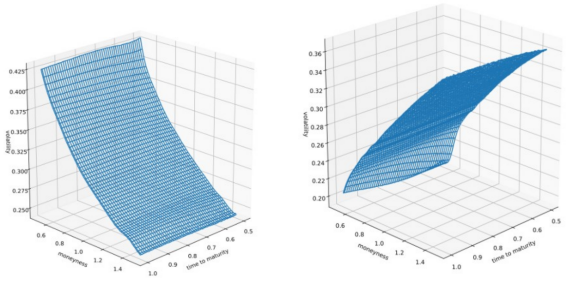


Figure 12: L^2 errors of CEV ($\beta = 3$) of PINN: price(left top), delta(right top), gamma(left bottom), theta(right bottom)



(a) $\beta = 1$ (b) $\beta = 3$

Figure 13: Volatility surfaces of CEV ($\beta = 1$) and ($\beta = 3$) via PINN

2.3.3 Volatility surface model

We have used the volatility surface in Figure 1 as the local volatility function. Figure 14 shows price), delta, gamma and theta of European put option with the volatility surface approximated by the neural network. Comparing GBM and CEV models, gamma is the lower, and theta is the higher when time t is 0. Figure 15 shows the local volatility surface obtained by (2.8).

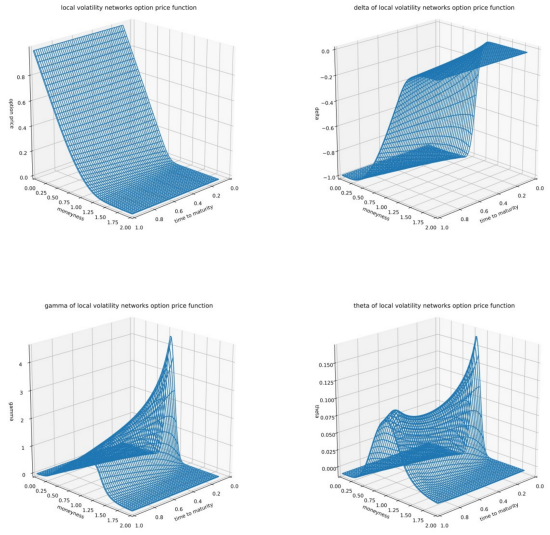


Figure 14: Volatility surface PINN approximation of European put option price and Greeks: price(left top), delta(right top), gamma(left bottom) and theta(right bottom)

3 ELS pricing with PINN

3.1 2 Asset step-down ELS

As a representative product of ELS, step-down ELS has a structure in which the suggested rate of return is obtained when certain early redemption conditions are met for each autocall early redemption evaluation date during the investment period, and the early redemption conditions are lowered in a cascade. And there is a Knock-In Barrier in the range lower than the early redemption condition, and even if the early redemption condition is not met before maturity, if the value of the underlying asset has never fallen below the knock-in barrier, the proposed rate of return can be obtained. Even when the value of the underlying assets has fallen below the knock-in barrier during the investment period, if all underlying assets rebound to the extent that the redemption conditions are met before maturity, the principal amount and the proposed rate of return can be obtained.

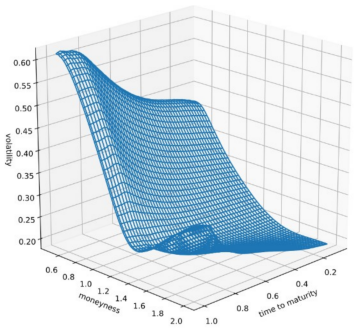


Figure 15: Local volatility surface obtained by PINN (2.8)

It is important to calculate the price and Greeks of ELS because they have a significant impact on decision-making of investor and hedging strategy of issuer. In [14], OSM is used to solve the multi-dimensional BSE and to calculate the price of ELS. As PINN has been widely used to solve PDEs, it has also been used in option pricing (refer to []). In this paper, we apply PINN to the multi-dimensional BSE and calculate the price and Greeks of a two-asset step-down ELS. Then, we compare the price and Greeks by PINN with those by OSM.

We selected example product to evaluate two-asset step-down ELS. The product is Korea Investment & Securities TRUE ELS 14361. It has two underlying assets and step down payoff structure.

3.1.1 Overview of 2 Asset step-down ELS

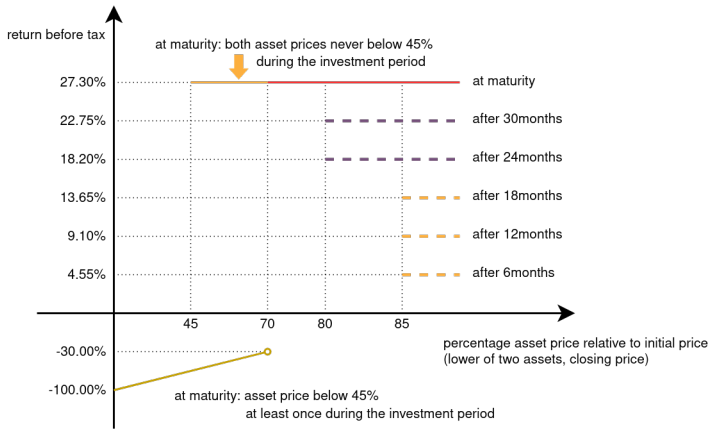


Figure 16: Payoff of Korea Investment & Securities TRUE ELS 14361

Figure 16 is a payoff structure of example product. The explanation of this ELS product is the following:

1. At each autocall early redemption evaluation date, if all of the prices of the underlying assets are greater than or equal to the strike price of the date, then the face value and the predetermined coupons are paid to a investor.

2. At the maturity date, if all of the prices of the underlying assets are greater than or equal to the strike price of the date, then the face value and the predetermined coupons are paid to a investor.

3. At the maturity date, when the above 2 is not satisfied, if all of the prices of the underlying assets haven't been less than the knock-in barrier even once before the maturity date, then the face value and the dummy are paid to a investor.

4. At the maturity date, when the above 2 and 3 are not satisfied, if at least one of the prices of the underlying assets have been less than the knock-in barrier even once before the maturity date, then the face value multiplied by (1 - rate of return of the most declined underlying asset) is paid to a investor.

Parameter	value
Underlying asset price $S_1(t_0)$	100
Underlying asset price $S_2(t_0)$	100
Volatility σ_1	28.6%
Volatility σ_2	54.7%
Correlation coefficient ρ	0.5371
Dividend yield d_1	0
Dividend yield d_2	0
Risk-free rate r_f	0.45%
Face value F	100
Strike $K_i, (i = 1, \ldots, 6)$	85%, 85%, 85%, 80%, 80%, 70%
Coupon rate $c_i, (i = 1, \ldots, 6)$	9.1% (Annual)
Dummy d	27.3%
Knock-In Barrier B	45%
Issue Date t_0	21.09.17
1st Evaluation Date t_1	22.03.14
2nd Evaluation Date t_2	22.09.13
3rd Evaluation Date t_3	23.03.14
4th Evaluation Date t_4	23.09.12
5th Evaluation Date t_5	24.03.12
Maturity Date t_6	24.09.09

Table 5: Parameters and Issuance information of ELS

In this section, we use parameters in Table 5. We assume that volatilities of underlying assets and risk-free rate are constant, and there is no dividend, and each evaluation date is the same as the each redemption date. With respect to time parameter, we assumed that t is the actual number of days in the period from one date to another date divided by 365, and we define from t_0 to t_6 as the actual number of days in the period from issue date to issue date and each evaluation date.

3.1.2 Initial and boundary conditions for ELS

To evaluate 2 Asset step-down ELS, we solve the two-dimensional BSE. The ELS has different prices when the knock-in barrier is touched and when it is not touched. Therefore, we consider initial and boundary conditions and some several conditions with two cases. One is when the underlying asset price does not touch the knock-in barrier, and the other is when the underlying asset price touches the knock-in barrier. The two-dimensional BSE with initial and boundary conditions for pricing 2 Asset step-down

ELS is as follows:

$$\begin{aligned}\frac{\partial V}{\partial t}+\frac{1}{2}\sigma_1^2 S_1^2(t)\frac{\partial^2 V}{\partial S_1^2}+\frac{1}{2}\sigma_2^2 S_2^2(t)\frac{\partial^2 V}{\partial S_2^2}+\rho\sigma_1\sigma_2 S_1(t)S_2(t)\frac{\partial^2 V}{\partial S_1\partial S_2}\\ +rS_1(t)\frac{\partial V}{\partial S_1}+rS_2(t)\frac{\partial V}{\partial S_2}=rV.\end{aligned} \tag{3.1}$$

Initial condition for $V(S_1, S_2, T)$:

$$V(S_1,S_2,T)_{\text{No touch}}=\begin{cases}(1+c_6)\times F & \text{if } \min\left(\frac{S_1(T)}{S_1(t_0)},\frac{S_2(T)}{S_2(t_0)}\right)\geq K_6,\\ (1+d)\times F & \text{if } K_6>\min\left(\frac{S_1(T)}{S_1(t_0)},\frac{S_2(T)}{S_2(t_0)}\right)\geq B,\\ \min\left(\frac{S_1(T)}{S_1(t_0)},\frac{S_2(T)}{S_2(t_0)}\right)\times F & \text{if } \min\left(\frac{S_1(T)}{S_1(t_0)},\frac{S_2(T)}{S_2(t_0)}\right)<B,\end{cases}$$

$$V(S_1,S_2,T)_{\text{touch}}=\begin{cases}(1+c_6)\times F & \text{if } \min\left(\frac{S_1(T)}{S_1(t_0)},\frac{S_2(T)}{S_2(t_0)}\right)\geq K_6,\\ \min\left(\frac{S_1(T)}{S_1(t_0)},\frac{S_2(T)}{S_2(t_0)}\right)\times F & \text{if } \min\left(\frac{S_1(T)}{S_1(t_0)},\frac{S_2(T)}{S_2(t_0)}\right)<K_6,\end{cases}$$

where $V(S_1, S_2, t)$ is the price of ELS and other parameters are in Table 5. The maximum value of S_1, S_2 is defined by L, we assume $L = 300$ in this section. The available ranges of S_1, S_2 are $0 \leq S_1, S_2 \leq L$. As the boundary conditions, we consider the linear boundary conditions when $S_1 = 0$ or $S_2 = 0$ and when $S_1 = L$ or $S_2 = L$.

Boundary condition

$$\begin{aligned}&\frac{\partial^2 V(0,S_2,t)_{\text{No touch}}}{\partial S_1^2}=0, \quad \frac{\partial^2 V(S_1,0,t)_{\text{No touch}}}{\partial S_2^2}=0,\\ &\frac{\partial^2 V(0,S_2,t)_{\text{touch}}}{\partial S_1^2}=0, \quad \frac{\partial^2 V(S_1,0,t)_{\text{touch}}}{\partial S_2^2}=0,\\ &\frac{\partial^2 V(L,S_2,t)_{\text{No touch}}}{\partial S_1^2}=0, \quad \frac{\partial^2 V(S_1,L,t)_{\text{No touch}}}{\partial S_2^2}=0,\\ &\frac{\partial^2 V(L,S_2,t)_{\text{touch}}}{\partial S_1^2}=0, \quad \frac{\partial^2 V(S_1,L,t)_{\text{touch}}}{\partial S_2^2}=0.\end{aligned}$$

Additionally, we consider payoff of autocall early redemption dates. At each evaluation date, ELS is redeemed with paying predetermined profit to investors. Therefore, the price of ELS is sum of the face value and the predetermined coupon when the condition of autocall early redemption is satisfied.

Autocall early redemptions for $i = 1, \ldots, 5$,

$$V(S_1, S_2, t_i)_{\text{No touch}} = (1 + c_i) \times F, \quad \text{if } \min\left(\frac{S_1(t_i)}{S_1(t_0)}, \frac{S_2(t_i)}{S_2(t_0)}\right) \geq K_i,$$

$$V(S_1, S_2, t_i)_{\text{touch}} = (1 + c_i) \times F, \quad \text{if } \min\left(\frac{S_1(t_i)}{S_1(t_0)}, \frac{S_2(t_i)}{S_2(t_0)}\right) \geq K_i.$$

Finally, when at least one of the underlying asset price is below the knock-in barrier, we consider one more condition to make the price of ELS when the knock-in barrier is not touched the same value with the price when the knock-in barrier is touched:

$$V(S_1, S_2, t)_{\text{No touch}} = V(S_1, S_2, t)_{\text{touch}} \quad \text{if } \min\left(\frac{S_1(t_i)}{S_1(t_0)}, \frac{S_2(t_i)}{S_2(t_0)}\right) < B.$$

3.2 Operator splitting method

OSM reduces multi-dimensional equation into multiple one-dimensional problems. We can rewrite (3.1) with respect to time to maturity $\tau = T - t$,

$$\begin{aligned} \frac{\partial V}{\partial \tau} &= \frac{1}{2}\sigma_1^2 S_1^2 \frac{\partial^2 V}{\partial S_1^2} + \frac{1}{2}\sigma_2^2 S_2^2 \frac{\partial^2 V}{\partial S_2^2} + \rho\sigma_1\sigma_2 S_1 S_2 \frac{\partial^2 V}{\partial S_1 \partial S_2} \\ &+ rS_1 \frac{\partial V}{\partial S_1} + rS_2 \frac{\partial V}{\partial S_2} - rV. \end{aligned} \tag{3.2}$$

Let splitting operators with respect to S_1 and S_2 denoted by $\mathcal{L}_{OS}^{S_1} V$, $\mathcal{L}_{OS}^{S_2} V$, and let $V_{ij}^n := V(S_{1i}, S_{2j}, \tau_n) = V(x_i, y_j, \tau_n)$, h is the length of space intervals, where S_{1i}, S_{2j} are treated as space variables $x_i := ih, y_j := jh$, respectively, and $\tau_n := n\Delta\tau$. Then, we obtain the following systems:

$$\begin{aligned} &\frac{V_{ij}^{n+\frac{1}{2}} - V_{ij}^n}{\Delta\tau} = \mathcal{L}_{OS}^x V^{n+\frac{1}{2}} \\ &:= \frac{1}{2}\sigma_x^2 x^2 \frac{V_{i-1,j}^{n+\frac{1}{2}} - 2V_{i,j}^{n+\frac{1}{2}} + V_{i+1,j}^{n+\frac{1}{2}}}{h^2} + rx_i \frac{V_{i+1,j}^{n+\frac{1}{2}} - V_{i-1,j}^{n+\frac{1}{2}}}{2h} - \frac{r}{2} V_{i,j}^{n+\frac{1}{2}} \\ &\quad + \frac{1}{2}\rho\sigma_x\sigma_y x_i y_j \frac{V_{i+1,j+1}^n + V_{i-1,j-1}^n - V_{i-1,j+1}^n - V_{i+1,j-1}^n}{4h^2}, \end{aligned} \tag{3.3}$$

$$\begin{aligned} &\frac{V_{ij}^{n+1} - V_{ij}^{n+\frac{1}{2}}}{\Delta\tau} = \mathcal{L}_{OS}^y V^{n+1} \\ &:= \frac{1}{2}\sigma_y^2 y^2 \frac{V_{i,j-1}^{n+1} - 2V_{i,j}^{n+1} + V_{i,j+1}^{n+1}}{h^2} + ry_j \frac{V_{i,j+1}^{n+1} - V_{i,j-1}^{n+1}}{2h} - \frac{r}{2} V_{i,j}^{n+1} \\ &\quad + \frac{1}{2}\rho\sigma_x\sigma_y x_i y_j \frac{V_{i+1,j+1}^{n+\frac{1}{2}} + V_{i-1,j-1}^{n+\frac{1}{2}} - V_{i-1,j+1}^{n+\frac{1}{2}} - V_{i+1,j-1}^{n+\frac{1}{2}}}{4h^2}. \end{aligned} \tag{3.4}$$

In OS schemes, difference with respect to time is sum of splitting operator $\mathcal{L}_{OS}^{S_1}V$, $\mathcal{L}_{OS}^{S_2}V$,

$$\frac{V_{ij}^{n+1} - V_{ij}^{n}}{\Delta\tau} = \mathcal{L}_{OS}^{x}V^{n+\frac{1}{2}} + \mathcal{L}_{OS}^{y}V^{n+1}. \tag{3.5}$$

Using the tridiagonal matrix algorithm, we calculate (3.3) in x-direction and (3.4) in y-direction sequentially. Then (3.5) is obtained. Repeating this procedure as many times as the number of the time meshes, we calculate the price of the ELS.

3.2.1 Tridiagonal matrix algorithm

Tridiagonal matrix algorithm also known as TDMA or Thomas algorithm is a simplified form of Gaussian elimination that can be used to solve tridiagonal systems of equations. For such systems, the solution can be obtained in $O(n)$ operations instead of $O(n^3)$ required by Gaussian elimination. A tridiagonal system for n unknowns can be written as

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i, \qquad \text{where } a_1 = 0, \text{ and } c_n = 0.$$

Algorithm 2 Tridiagonal matrix algorithm

Input: Vectors a, b, c, d, and length of vectors n.
Output: Unknown vector x.

Forward elimination phase.
for $i \in 2$ to n **do**
  $w = \frac{a[i]}{b[i-1]}$,
  $b[i] = b[i] - w \times c[i-1]$,
  $d[i] = d[i] - w \times d[i-1]$.
end for
Backward substitution phase.
$x[n] = \frac{d[n]}{b[n]}$.
for $i \in n-1$ to 1 **do**
  $x[i] = \frac{d[i] - c[i] \times x[i+1]}{b[i]}$,
end for

3.3 PINN for ELS pricing

To apply PINN to (3.1), we need to set up the optimization problem.

3.3.1 Scale invariance of Black-Scholes model

Before setting up the optimization problem, we need to adjust the scale of problem. When training PINN, it shows poor performance if inputs are large value without scaling. To train PINN, we adjust the underlying asset price by dividing with 100, and we also adjust the price of ELS by dividing with 100. Because the price of ELS has the same scale with the price of underlying assets, we esimate the scaled ELS price and multiply by 100.

We can show that (3.1) is scale invariant. We rewrite (3.2) setting $u(\tau, x, y) := V(\tau, S_1, S_2)$:

$$\frac{\partial u}{\partial \tau} - \frac{1}{2}\sigma_x^2 x^2 \frac{\partial^2 u}{\partial x^2} - \frac{1}{2}\sigma_y^2 y^2 \frac{\partial^2 u}{\partial y^2} - \rho\sigma_x\sigma_y xy \frac{\partial^2 u}{\partial x \partial y} - rx\frac{\partial u}{\partial x} - ry\frac{\partial u}{\partial y} + ru = 0. \quad (3.6)$$

From (3.6), we obtain the following scaled equation $v(\tau, \hat{x}, \hat{y})$

$$\frac{\partial v}{\partial \tau} - \frac{1}{2}\sigma_x^2 \hat{x}^2 \frac{\partial^2 v}{\partial \hat{x}^2} - \frac{1}{2}\sigma_y^2 \hat{y}^2 \frac{\partial^2 v}{\partial \hat{y}^2} - \rho\sigma_x\sigma_y \hat{x}\hat{y} \frac{\partial^2 v}{\partial \hat{x} \partial \hat{y}} - r\hat{x}\frac{\partial v}{\partial \hat{x}} - r\hat{y}\frac{\partial v}{\partial \hat{y}} + rv = 0. \quad (3.7)$$

In general, if the option price $u(\tau, x, y)$ satisfies (3.6) for all $\tau \in [0, T], x \in [0, M], y \in [0, M]$ with the initial condition $u(0, x, y) = \psi(x, y)$, where x, y are the underlying asset price, and M is a relatively large number, then (3.7) holds for the scaled option price $v(\tau, \hat{x}, \hat{y}) = \frac{u(\tau, x, y)}{N}$ with the initial condition $v(0, \hat{x}, \hat{y}) = \frac{\psi(x, y)}{N}$, where the scaled variables $\hat{x} = \frac{x}{N} \in [0, \frac{M}{N}], \hat{y} = \frac{y}{N} \in [0, \frac{M}{N}]$, where N is the real number. The parameters $\tau, r, \sigma_x, \sigma_y, \rho$ in (3.7) are the same as those in (3.6).

3.4 Optimization problem

In this subsection, we define an objective function and loss functions and train PINN with random spatial points $(\tau, x, y) \in [0, T] \times [0, 3] \times [0, 3]$, where τ is time to maturity, x, y are the scaled underlying asset prices.

Loss functions for $i = 1, 2$,

$$\mathcal{L}_{pde}(u_i^\theta) := \Big\| \frac{\partial u_i^\theta(\tau,x,y)}{\partial \tau} - \frac{1}{2}\sigma_x^2 x^2 \frac{\partial^2 u_i^\theta(\tau,x,y)}{\partial x^2} - \frac{1}{2}\sigma_y^2 y^2 \frac{\partial^2 u_i^\theta(\tau,x,y)}{\partial y^2}$$
$$- \rho\sigma_x\sigma_y xy \frac{\partial^2 u_i^\theta(\tau,x,y)}{\partial x \partial y} - rx\frac{\partial u_i^\theta(\tau,x,y)}{\partial x} - ry\frac{\partial u_i^\theta(\tau,x,y)}{\partial y} + ru_i(\tau,x,y)\Big\|_2^2,$$
$$\mathcal{L}_{ic}(u_i^\theta) := \|u_i^\theta(0,x,y) - f_i(x,y)\|_2^2,$$
$$\mathcal{L}_{bc}(u_i^\theta) := \Big\| \frac{\partial^2 u_i^\theta(\tau,0,y)}{\partial x^2} \Big\|_2^2 + \Big\| \frac{\partial^2 u_i^\theta(\tau,x,0)}{\partial y^2} \Big\|_2^2 + \Big\| \frac{\partial^2 u_i^\theta(\tau,3,y)}{\partial x^2} \Big\|_2^2 + \Big\| \frac{\partial^2 u_i^\theta(\tau,x,3)}{\partial y^2} \Big\|_2^2,$$
$$\mathcal{L}_{ac}(u_i^\theta) := \sum_{j=1}^{5} \|u_i^\theta(\tau_j,x,y) - (1+c_j)\|_2^2,$$
$$\mathcal{L}_{ki}(u_1^\theta) := \|u_1^\theta(\tau,x,y) - u_2^\theta(\tau,x,y)\|_2^2,$$

where

$$f_1(x,y) = \begin{cases} 1+c_6 & \text{if } \min(x,y) \geq K_6, \\ 1+d & \text{if } K_6 > \min(x,y) \geq B, \\ \min(x,y) & \text{if } \min(x,y) < B, \end{cases}$$
$$f_2(x,y) = \begin{cases} 1+c_6 & \text{if } \min(x,y) \geq K_6, \\ \min(x,y) & \text{if } \min(x,y) < K_6. \end{cases}$$

Objective function

No touch the knock-in barrier:

$$\mathcal{L}_{total}(u_1^\theta) = \mathcal{L}_{pde}(u_1^\theta) + \mathcal{L}_{ic}(u_1^\theta) + \mathcal{L}_{bc}(u_1^\theta) + \mathcal{L}_{ac}(u_1^\theta) + \mathcal{L}_{ki}(u_1^\theta),$$

Touch the knock-in barrier:

$$\mathcal{L}_{total}(u_2^\theta) = \mathcal{L}_{pde}(u_2^\theta) + \mathcal{L}_{ic}(u_2^\theta) + \mathcal{L}_{bc}(u_2^\theta) + \mathcal{L}_{ac}(u_2^\theta),$$

where u_1^θ and u_2^θ denote the scaled option price estimated by PINN with no touch the knock-in barrier and with touch the knock-in barrier, respectively, and θ are parameters of neural network.

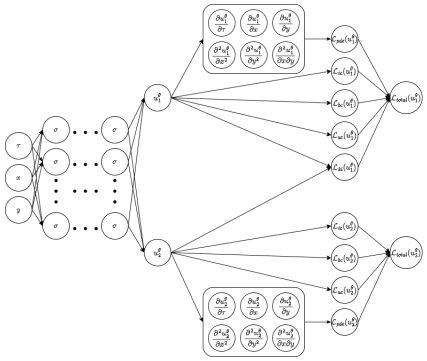


Figure 17: Schematic diagram of PINN for pricing ELS.

Algorithm 3 Algorithm for PINN

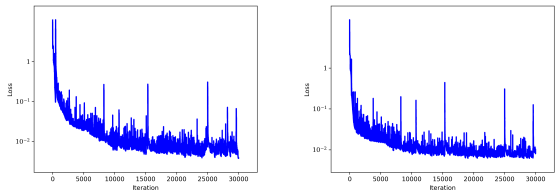
Input: Vector of random spatial points $(\tau, x, y) \in [0, T] \times [0, 3] \times [0, 3]$.
Output: Vector of parametric solution of two-dimensional BSE u_1^θ, u_2^θ.

```
Initialize epoch as 0.
while (L_total(u_1^θ) > 1e−5 or L_total(u_2^θ) > 1e−5) or epoch < 150 do
    Initialize step as 0.
    Uniformly generate N_ic, N_ac, N_bc, N_pde, N_ki
    while (L_total(u_1^θ) > 1e−6 or L_total(u_2^θ) > 1e−6) or step < 200 do
        Calculate the total loss functions L_total(u_1^θ), L_total(u_2^θ).
        Update θ by back propagation algorithm.
        Add 1 to step.
    end while
    Add 1 to epoch.
end while
```

Figure 17 is the schematic diagram of PINN for pricing ELS. As hyper parameters, we use the Adam optimizer with learning rate of 0.001, and 7 hidden layers. Each hidden layer has 256 neurons, and softplus is used as a activation function. Weights of each layer are initialized by Xavier initialization. We consider batch size $N_{ic} = 500, N_{ac} = 500, N_{bc} = 400, N_{pde} = N_{ic} + N_{ac} + N_{bc} + 10000$. And we consider N_{ki} is the part of N_{pde}, which contains x or y that is less than B, where $N_{ic}, N_{ac}, N_{bc}, N_{pde}, N_{ki}$ are the numbers of random spatial points to train PINN. For generalization performance of the model, in all hidden layers, before the activation function operation is performed, batch normalization is performed. And we randomly generate spatial points for one epoch, and the points are discarded and newly generated for every epochs. The algorithm of PINN is Algorithm 3.

3.5 Numerical Simulation

We calculate the price of ELS and Greeks by implementing PINN, OSM using CPU(12th Gen Intel(R) Core(TM) i9-12900KF), GPU(NVIDIA Corporation GA102 [GeForce RTX 3090] (rev a1)), and Pytorch.



(a) Total loss for No touch. (b) Total loss for Touch.

Figure 18: Total losses for PINN depending on the touch of knock-in barrier

Figure 18 shows change of total loss during the algorithm iteration. Loss for ELS when knock-in barrier is not touched is 6.2970e-3, and Loss for ELS when knock-in barrier is touched is 9.0489e-3. Given the hyper parameters we use, the value of loss appears to be fluctuated because we newly generate and use the random spatial points for every epochs.

3.5.1 Price

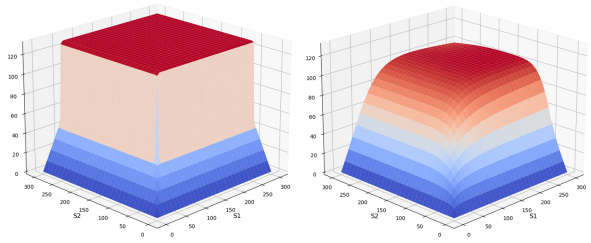
The result of pricing ELS is as follows:

	PINN (150 Epochs $\times$ 200 steps)	OSM (Mesh: 61 $\times$ 61)
Price($)	90.9845	88.1280
Time(sec)	3793.3523	12.4759

The real value of fair price in the investment prospectus is 85.5074. The prices obtained by OSM are similar to each other, and they are slightly larger than the real value. It is presumed that parameters and assumptions we use would be different from those used in reality. The price by PINN appears similar to the price by OSM, but it is relatively larger than the real value and the prices by OSM. It is presumed that the reason of this phenomenon might be that PINN is not trained enough with iteration of 30000 times given hyper parameters.

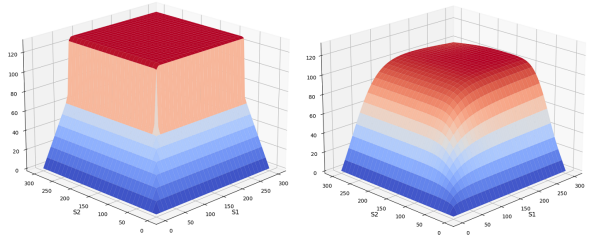
Figures 19, 20 are prices calculated by OSM, and Figures 21, 22 by PINN. And the prices by PINN and OSM seem that they are symmetrically

transformed by the axis of $S_1 = S_2$. Based on the results, the prices by PINN at $\tau = 0$ do not catch the payoff well.



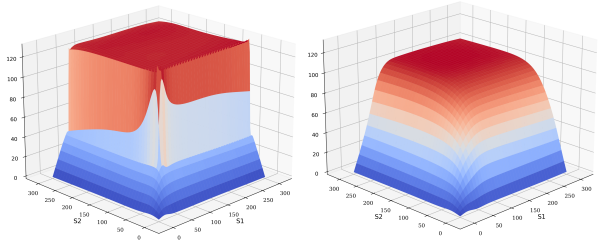
(a) ELS price at $\tau = 0$. (b) ELS price at $\tau = T$.

Figure 19: ELS prices by OSM when barrier is not touched



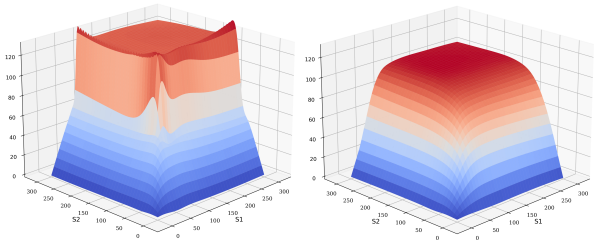
(a) ELS price at $\tau = 0$. (b) ELS price at $\tau = T$.

Figure 20: ELS prices by OSM when barrier is touched



(a) ELS price at $\tau = 0$. (b) ELS price at $\tau = T$.

Figure 21: ELS prices by PINN when barrier is not touched

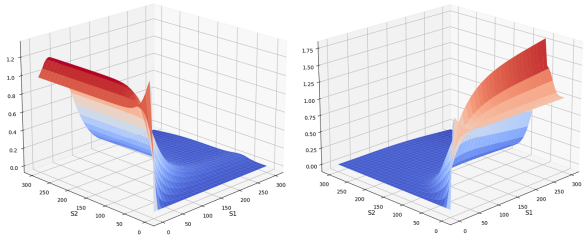


(a) ELS price at $\tau = 0$. (b) ELS price at $\tau = T$.

Figure 22: ELS prices by PINN when barrier is touched

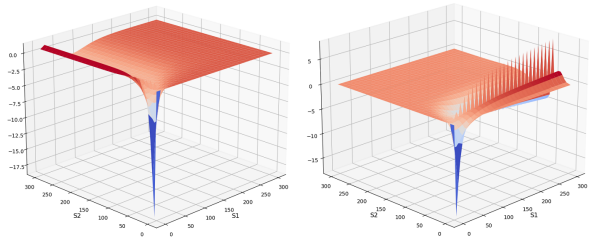
3.5.2 Greeks

Figures 23, 24 and 25 are the Greeks calculated by OSM, and Figures 26, 27 and 28 are the Greeks calculated by PINN. The Greeks calculated by OSM appears smooth but unstable in some areas where the value of Greeks steeply changes. In the case of PINN, the Greeks by PINN appears similar to the Greeks by OSM. And they seem like that they are symmetrically transformed by the axis of $S_1 = S_2$, where the phenomenon is similar to the that of the price by PINN.



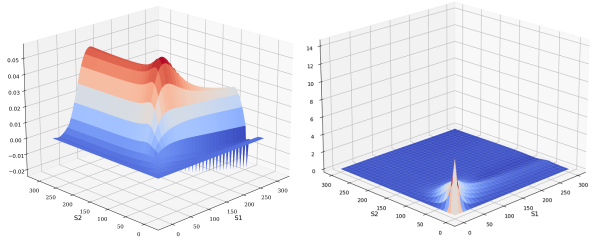
(a) Delta 1 by OSM (b) Delta 2 by OSM

Figure 23: Delta by OSM when barrier is not touched



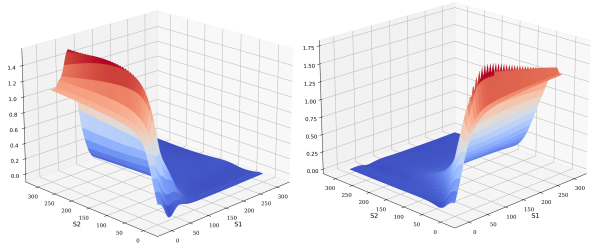
(a) Gamma 1 by OSM (b) Gamma 2 by OSM

Figure 24: Gamma by OSM when barrier is not touched



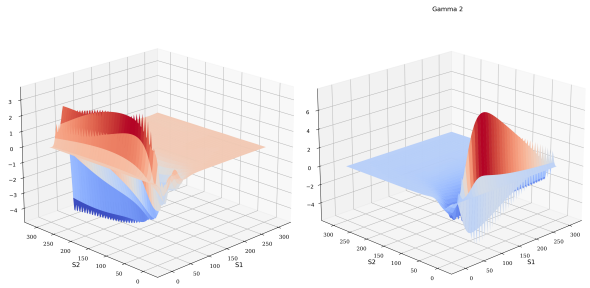
(a) Theta by OSM (b) Cross gamma

Figure 25: Theta and Cross gamma by OSM when barrier is not touched



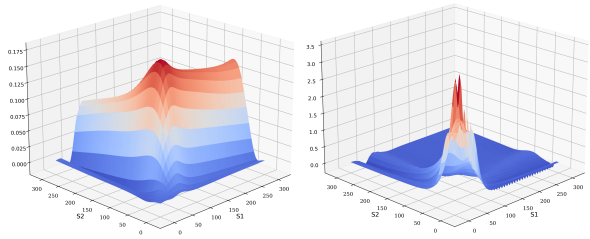
(a) Delta 1 by PINN (b) Delta 2 by PINN

Figure 26: Delta by PINN when barrier is not touched



(a) Gamma 1 by PINN (b) Gamma 2 by PINN

Figure 27: Gamma by PINN when barrier is not touched



(a) Theta by PINN (b) Cross gamma by PINN

Figure 28: Theta and Cross gamma by PINN when barrier is not touched

4 Conclusion

Deep learning has received a lot of attention due to its high performance. We focus on the power of artificial neural networks as function approximations. We have implemented neural networks to solve parametric BSEs under local volatility models. We propose algorithms to improve the performance of BSE approximation. Data random generation scheme is used for approximate performance. The transfer learning in Section 2.2.2 improves the training speed and training convergence of the models. As a result in Section 2.3, the approximate performance of local volatility models was evaluated through closed-form solutions and Dupire's equation. Prices and Greeks are approximated well in measurement by solutions. It is verified by Dupire's equation that the parametric BSE is well approximated. Pricing and hedging derivatives under the local volatility is important to practitioners. It is expected

that this study can contribute to practitioners as an analysis tool of the local volatility model.

We have also implemented PINN to solve the two-dimensional BSE for pricing two-asset step-down ELS, and compared with OSM. PINN has advantage of computing partial derivatives easily by automatic differentiation. And PINN for two-dimensional BSE shows similar price to the prices by OSM, and similar Greeks to the Greeks by OSM. However, it's accuracy is less than OSM it requires larger computational cost than OSM. To improve the accuracy in a reasonable time, hyper parameters such as an activation function which fits the best for the two-dimensional BSE needs to be improved. And methods for reducing computational cost are also need to be improved.

Acknowledgement: Bae (NRF-2021R1A2C109338) have been partially supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology.

References

[1] Y. Bai, T. Chaolu and S. Bilige, *The application of improved physics-informed neural network (IPINN) method in finance*, Nonlinear Dynamics Vol. 107 (2022), no.4, 3655 - 3667

[2] J. Berner, M. Dablander and P. Grohs, *Numerically solving parametric families of high-dimensional Kolmogorov partial differential equations via deep learning*, Advances in Neural Information Processing Systems Vol.33 (2020), 16615-16627

[3] F. Black and M. Scholes, *The Pricing of Options and Corporate Liabilities*, J. Political Economy Vol. 81 (1973), no.3, 637-54,

[4] P. Carr and R. Lee, *Volatility derivatives*, Annu. Rev. Financ. Econ., Vol. 1:319-339 (Volume publication date 2009) https://doi.org/10.1146/annurev.financial.050808.114304

[5] T.F. Coleman, Y. Li and A. Verma, *Reconstructing the unknown local volatility function*, Quantitative Analysis in Financial Markets, (2001), 192-215

[6] J.C. Cox and S.A. Ross, *The valuation of options for alternative stochastic processes*, J. financial economics Vol. 3 (1976), no.1-2, 145-166

[7] J.C. Cox, *Notes on option pricing I: Constant elasticity of variance diffusions*, Unpublished note, Stanford University, Graduate School of Business (1975)

[8] E. Derman and I. Kani, *Riding on a smile*, Risk Vol. 7, (1994), no.2, 32-39

[9] B. Dupire, *Pricing with a Smile*, Risk Magazine (1994), 18-20

[10] D.C. Emanuel and J.D. MacBeth, *Further results on the constant elasticity of variance call option pricing model*, J. Financial and Quantitative Analysis Vol. 17 (1982), no.4, 533-554

[11] J. Gatheral, The Volatility Surface: A Practictioner's Guide, 2011, John Wiley and Sons, Inc.

[12] G. Cybenko, *Approximation by superpositions of a sigmoidal function*, Mathematics of control, signals and systems Vol. 2 (1989), no.2, 303-314

[13] K. Glau and L. Wunderlich, *The deep parametric PDE method and applications to option pricing*, Applied Mathematics and Computation Vol. 432 (2022), 127355

[14] D.-R. Jeong, I.-S. Wee and J.-S. Kim, *An operator splitting method for pricing the ELS option*, J. of the Korean Society for Industrial and Applied Mathematics Vol. 14 (2010), no.3 175-187

[15] S. Kang, ELS pricing with Physics-Informed Neural Network, Master Thesis, 2023, Ajou University.

[16] Y. Kim, H.-O. Bae and H.K. Koo, *Option pricing and Greeks via a moving least square meshfree method*, Quantitative Finance Vol. 14 (2014), no.10, 1753-1764

[17] I.E. Lagaris, A. Likas and D.I. Fotiadis, *Artificial neural networks for solving ordinary and partial differential equations*, IEEE transactions on neural networks Vol. 9 (1998) no.5, 987-1000

[18] M. Larguinho, J.C. Dias and C.A. Braumann, *On the computation of option prices and Greeks under the CEV model*, Quantitative Finance Vol. 13 (2013), no.6, 907-917.

[19] H. Lee and I.S. Kang, *Neural algorithm for solving differential equations*, J. Computational Physics Vol. 91 (1990), no.1, 110-131

[20] M. Lee, Solving Black-Scholes PDE Associated with Local Volatility via Physics-Informed Neural Network, Master Thesis, 2022, Ajou University.

[21] H. Lim and H.-O. Bae, *Construction of the Implied Volatility Surface by Thin Plate Spline Function*, Korean J. of Financial Engineering Vol. 18 (2019), no.4, 1-36, 10.35527/kfedoi.2019.18.4.001

[22] A.J. Meade Jr and A.A. Fernandez, *Solution of nonlinear ordinary differential equations by feedforward neural networks*, Mathematical and Computer Modelling Vol. 20 (1994), no.9, 19-44

[23] M. Raissi, P. Perdikariss and G.E. Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, J. Computational physics Vol. 378 (2019), 686-707

[24] J. Sirignano and K. Spiliopoulos, *DGM: A deep learning algorithm for solving partial differential equations*, J. computational physics Vol. 375 (2018), 1339-1364

[25] X. Wang, J. Li and J. Li, *A Deep Learning Based Numerical PDE Method for Option Pricing*, Computational Economics (2022), https://doi.org/10.1007/s10614-022-10279-x

[26] K. Woo, H.-O. Bae and Y. Kim, *Financial Derivatives Pricing under Stochastic Alpha Beta Rho(SABR) Model*, Korean J. of Financial Engineering Vol. 15 (2016), no.4, 1-27 10.35527/kfedoi.2016.15.4.001

[27] R. Yentis and M.E. Zaghloul, *VLSI implementation of locally connected neural network for solving partial differential equations*, IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications Vol. 43 (1996), no.8, 687-690