Solving Black-Scholes Equation with Physics Informed Neural Networks

Keon Lee
School of Computing
Korea Advanced Institute of Science and Technology

Abstract

Black-Scholes equation was a revolutionary discovery in the field of finance. Through this partial differential equation, the ratio of how the price change of the option relates to the price change of the underlying asset was able to be calculated. In this project, physics-informed neural networks(PINNs) was used to derive the numerical solutions for the Black-Sholes equation pricing European options.

1 Introduction

1.1 Financial Options

An option is a contract that grants the right to buy or sell an underlying asset within a given time frame and subject to certain conditions. The *exercise price* or *strike price* refers to the cost of the asset when the option is exercised. The *expiration date* or *maturity date* is the last day the option may be exercised.

Options are financial derivatives from underlying assets, including *call options* and *put options*. Call options refer to the right to buy at a predetermined price, while put options refer to the right to sell at a predetermined price. Exploiting price differences over time makes options derivatives that can be used by investors for hedging risks.

Currently in real world market, there are several different *styles* of options, according to the dates on which the option may be exercised. When an option can only be exercised at maturity, it is referred to as a European style; when it can be exercised on or before maturity, it is referred to as an American style. (Brealey et al., 2012) In South Korea, option trading follows the European style, and can only be exercised at maturity. Hence in this project we only focus on the European style options.

The profit of option sellers is maximised only when the price of the option is low, and the profit of option buyers is maximised only when the price of the option is high. Thus, the price for an option should be determined in order that both the seller and the buyer can agree. Otherwise, the possibility of causing damage to one side increases, making it an unfair price.

1.2 Black-Scholes Model

The theoretical estimate of the price of European style options can be determined using the Black-Scholes model, a mathematical representation of the dynamics of the financial market. (Higham, 2004; Oksendal, 2013) The Black-Scholes equation is derived under the following seven assumptions. (Black & Scholes, 1973)

1. The rate of return on the riskless asset is constant and thus called the risk-free interest rate.
2. The stock price follows a *geometric Brownian motion*, and it is assumed that the drift and volatility of the motion are constant.
3. The stock does not pay a dividend.

4. There is no way to make a riskless profit.
5. The market has the ability to borrow and lend any amount, even fractional, of cash at the riskless rate.
6. The market has the ability to buy and sell any amount, even fractional, of the stock.
7. The above transactions do not incur any fees or costs.

Following the above assumptions, the Black-Scholes equation is derived as (Black & Scholes, 1973)

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

where

$S(t)$	is the price of the underlying asset at time t (also denoted as S_t),
$V(t, S)$	is the price of the option as a function of the underlying asset S at time t,
r	is the annualised risk-free interest rate, continuously compounded, and
σ	is the standard deviation of the stock's returns, also known as volatility.

2 Problem Formulation

2.1 Approach

In order to find the solutions for the above partial differential equation, *physics informed nerual networks* will be used. PINNs provide the network model with established equations that control the physics of a system. (Raissi et al., 2019) This framework can be used to identify parameters and solve systems of partial differential equations and ordinary differential equations. (Haghighat et al., 2021) It is known that PINN can be used to identify an optimal solution with high fidelity if some physical understanding of the problem and some form of training data (even sparse and partial) are available. (Arzani et al., 2021)

The use of PINNs instead of analytic solutions or traditional numerical methods is preferred because of the high extensibility and expressivity of neural networks. For instance, the above problem can be extended to consider American style options, or options with dividend by changing the loss functions. Furthermore, without the requirement for retraining, the trained PINN network can be used to predict the values on simulation grids with various resolutions. (Markidis, 2021)

2.2 Boundary Conditions

One can utilise the Black–Scholes equation to calculate the price of European put and call options by substituting V. In such cases, the boundary conditions must be given.

Define variables and functions as follows.

t	is a time in years.
$C(t, S)$	is the price of a European call option.
$P(t, S)$	is the price of a European put option.
K	is the strike price of the option.
T	is the time of option expiration.

Let us consider the call option price for European style options.

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + rS\frac{\partial C}{\partial S} - rC = 0 \tag{2}$$

For European style call options, the boundary conditions are given as follows. (Black & Scholes, 1973)

$$C(t, 0) = 0 \qquad \text{for all } t \geq 0 \tag{3}$$

$$C(t, S) \to S - K \qquad \text{for all } t \geq 0 \text{ as } S \to \infty \tag{4}$$

$$C(T, S) = \max\{S - K, 0\} \tag{5}$$

2.3 Loss Functions

In this section, we will construct the loss functions to be used for the training of neural networks. Let $u(t, S)$ be the function predictions of the neural network for inputs S and t. Let

$$f(t, S) = \frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + rS\frac{\partial C}{\partial S} - rC$$

Now, we can define the loss functions as follows.

The collocation points are the subject of our first loss function.

$$\mathcal{L}_f = \frac{1}{N_f}\sum_{i=1}^{N_f} \left(f(t_i, S_i)\right)^2 \tag{6}$$

where

N_f	is the number of sampled collocation points, and
$\{(t_i, S_i)\}_{i=1}^{N_f}$	is the set of randomly sampled collocation points.

Similar work can be done for the derivative value at expiration.

$$\mathcal{L}_e = \frac{1}{N_e}\sum_{i=1}^{N_e} \left(u(T, S_i) - C(T, S_i)\right)^2 \tag{7}$$

where

N_e	is the number of sampled expiration points, and
$\{S_i\}_{i=1}^{N_f}$	is the set of randomly selected asset prices at expiration.

Finally, the loss function for boundary values can be defined as follows.

$$\mathcal{L}_b^{(1)} = \frac{1}{N_b^{(1)}}\sum_{i=1}^{N_b^{(1)}} \left(u(t_i, 0)\right)^2 \tag{8}$$

$$\mathcal{L}_b^{(2)} = \frac{1}{N_b^{(2)}}\sum_{i=1}^{N_b^{(2)}} \left(u(t_i, M) - C^*(t_i, M)\right)^2 \tag{9}$$

where

M	is the upper bound for the price of assets,
$N_b^{(1)}$	is the number of sampled boundary points at $S = 0$,
$N_b^{(2)}$	is the number of sampled boundary points at $S = M$,
$C^*(t, S)$	is the target value for time t and asset price S (will be elaborated later), and
$\{t_i\}_{i=1}^{N_f}$	is the set of randomly selected time points.

In total, we can define our objective function as a sum of above loss functions.

$$\mathcal{L} = \mathcal{L}_f + \mathcal{L}_e + \mathcal{L}_b$$

where $\mathcal{L}_b = \mathcal{L}_b^{(1)} + \mathcal{L}_b^{(2)}$.

Our neural network should be able to reliably predict numerical solutions to the Black-Scholes equation by minimising the aforementioned loss function. (Raissi & Karniadakis, 2018)

3 Implementation

We concentrate on an European call option on a non-dividend paying asset for our implementation example. For simplicity, the parameters of the Black-Scholes model is set as in table 1.

Table 1: Parameter settings for Black-Scholes model

PARAMETER	**VALUE USED**
Strike price (K)	40
Interest rate (r)	0.05
Annual volatility of asset (σ)	0.25
Time to maturity (T)	1
Input ranges for time	$[0, 1]$
Input ranges for asset price	$[0, 500]$

3.1 Data Generation

For input values of our physics infomed neural network, we need to generate data from three different domains. One is used as inputs for differential loss, another is used as inputs satisfying the boundary conditions of Black-Scholes equation, and the other is used inputs satisfying the values at expiration for the equation.

To begin with, let us consider the sampling of collocation points. Since the upper and lower bound for both asset price (S) and time (t) are set, random values between the bounds are sampled uniformly using `PyTorch` built-in functions.

$$\mathcal{X}_f = \{(t_i, S_i) : t_i \sim \mathcal{U}(0,1),\ S_i \sim \mathcal{U}(0,500)\}_{i=1}^{N_f} \tag{10}$$

$$\mathcal{Y}_f = \{y_i = 0\}_{i=1}^{N_f} \tag{11}$$

Similarly, sampling asset prices at expiration can be done using the boundary condition given in equation 5.

$$\mathcal{X}_e = \{(1, S_i) : S_i \sim \mathcal{U}(0,500)\}_{i=1}^{N_e} \tag{12}$$

$$\mathcal{Y}_e = \{y_i = \max\{S_i - K, 0\} : (1, S_i) \in \mathcal{X}_e\}_{i=1}^{N_e} \tag{13}$$

Sampling times at the boundary can be done in a similar manner; there are some considerations that should be made. Because the boundary condition in equation 4 holds only when $S \to \infty$ which is highly unrealistic assumption, we cannot set the boundary condition as $C(S, K) = S - K$. Instead, we will use the known analytic solutions for the Black-Scholes equation, which is given as follows. (Black, 1976)

$$C^*(T, S) = S\Phi(d_+) - Ke^{-rT}\Phi(d_-) \tag{14}$$

where

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} dt$$

(Cumulative distributive function of the standard 1D Gaussian distribution)

$$d_\pm = \frac{\ln(S/K) + (r \pm \sigma^2/2)T}{\sigma\sqrt{T}}$$

Now we can sample the data as follows.

$$\mathcal{X}_b^{(1)} = \{(t_i, 0) : t_i \sim \mathcal{U}(0,1)\}_{i=1}^{N_b^{(1)}} \tag{15}$$

$$\mathcal{Y}_b^{(1)} = \{y_i = 0\}_{i=1}^{N_b^{(1)}} \tag{16}$$

$$\mathcal{X}_b^{(2)} = \{(t_i, 500) : t_i \sim \mathcal{U}(0,1)\}_{i=1}^{N_b^{(2)}} \tag{17}$$

$$\mathcal{Y}_b^{(2)} = \{y_i = C^*(t_i, 500) : (t_i, 500) \in \mathcal{X}_b^{(2)}\}_{i=1}^{N_b^{(2)}} \tag{18}$$

Figure 1 shows a sampling example with $N_f = 5000, N_e = 1000, N_b = N_b^{(1)} = N_b^{(2)} = 1000$.

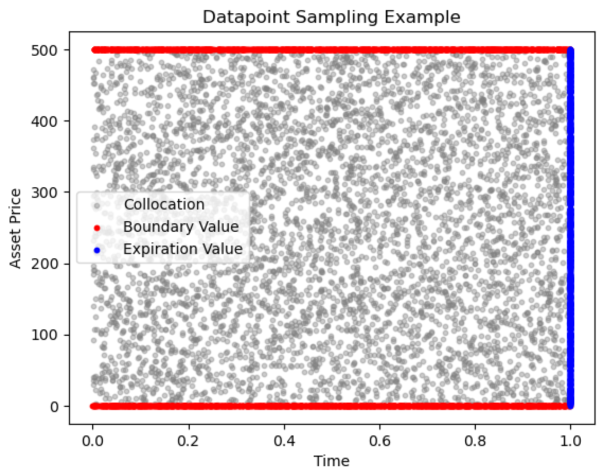


Figure 1: An example of sampling datapoints.

3.2 Model Architecture

The initial model built for this project consists of 5 hidden fully-connected layers. Each hidden layer has 128 neurons. Sigmoid function is used for activation and is placed after each fully connected layer. Figure 2 shows the overall architecture of the neural network model.

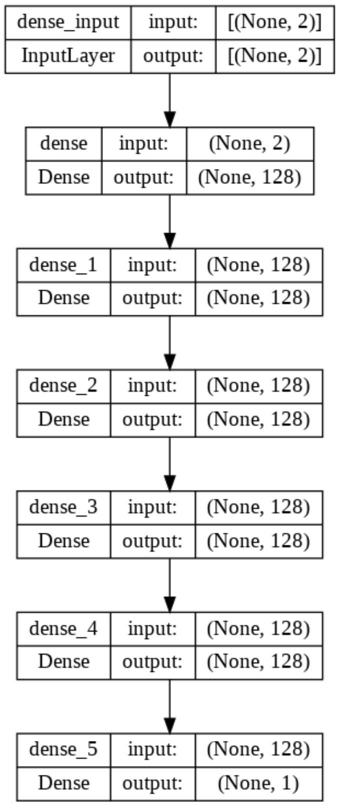

dense_input | input: | [(None, 2)]
InputLayer | output: | [(None, 2)]
dense | input: | (None, 2)
Dense | output: | (None, 128)
dense_1 | input: | (None, 128)
Dense | output: | (None, 128)
dense_2 | input: | (None, 128)
Dense | output: | (None, 128)
dense_3 | input: | (None, 128)
Dense | output: | (None, 128)
dense_4 | input: | (None, 128)
Dense | output: | (None, 128)
dense_5 | input: | (None, 128)
Dense | output: | (None, 1)

Figure 2: Model architecture used for project.

4 Experiments

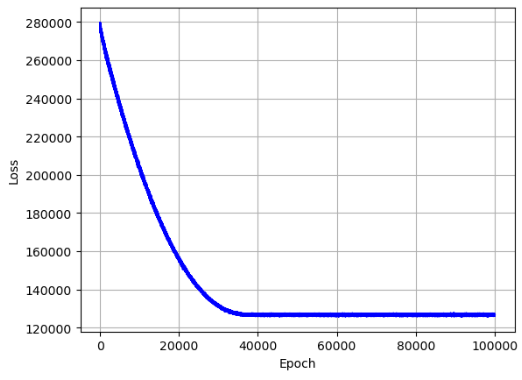
In this section, experimental results for the constructed PINN model are presented. First experiment is conducted with hyperparameters presented in table 2.

Table 2: Hyperparameter settings for initial experiment

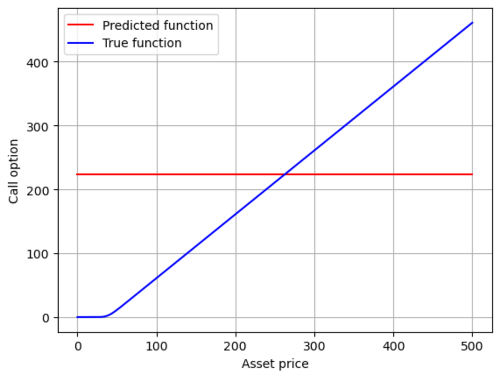
HYPERPARAMETER	**VALUE USED**
Learning Rate (η)	5.0×10^{-5}
Epochs	100,000
Number of sampled collocation points (N_f)	5000
Number of sample expiration points (N_e)	5000
Number of sample boundary points (N_b)	5000
Weight initialisation	Zero
Activation function	Sigmoid (σ)

The optimiser used for minimising the loss function in this project is *Adam*, which performs admirably and outperforms several other stochastic optimisation techniques. (Kingma & Ba, 2014)

The following figure 3a shows the change in the loss function according to the epoch. However, when the predicted function is compared against the true function, it showed significant difference. Figure 3b shows the comparison between the true function C^* and the predicted function u at $t = 0.5$.



(a) Change of loss function over epoch



(b) Comparison with true function

Figure 3: Results of the first experiment.

Therefore, some adjustments must be made in order to improve the overall quality of the training. First, the activation functions were changed to rectified linear unit (ReLU) function instead of sigmoid function, in order to mitigate gradient vanishing problems in deep neural networks, thus showing better convergence performance. (Krizhevsky et al., 2017) Table 3 shows the Hyperparameter settings for the second experiment.

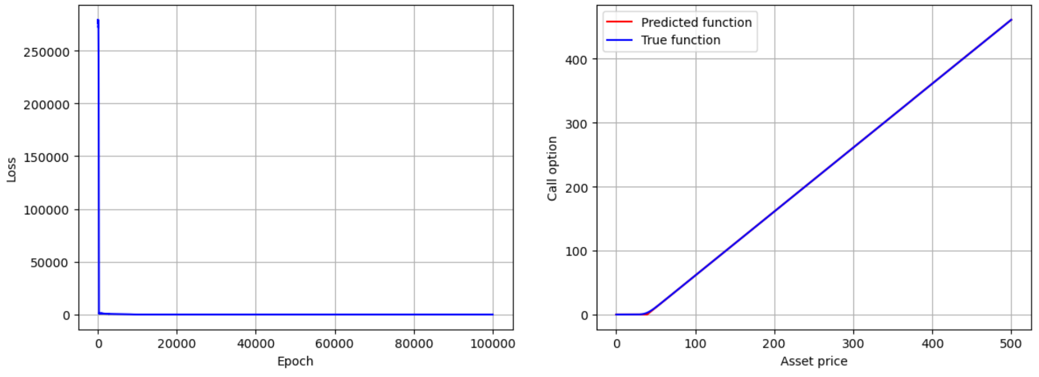
Figure 4a shows the improved results of the second experiment, and figure 4b shows the comparison between the predicted function and the true function, calculated at $t = 0.5$.

Another major adjustment made during the experiments of this project was to change the **weight initialisation method**, which was barely made in other previous studies or implementations utilising PINNs. It is well known that weight initialisation scheme can affect the overall performance of the neural network prediction by mitigating several gradient vanishing problems. (Géron, 2022) In fact, a neural network's convergence depends highly on the weights being initialised correctly. (Kumar, 2017)

One of the most famous choice for weight initialisation is *He* method, also known as *Kaiming* initialisation. Kaiming initialisation is a neural network initialisation technique that considers the

Table 3: Hyperparameter settings for second experiment

HYPERPARAMETER	**VALUE USED**
Learning Rate (η)	5.0×10^{-5}
Epochs	100,000
Number of sampled collocation points (N_f)	5000
Number of sample expiration points (N_e)	5000
Number of sample boundary points (N_b)	5000
Weight initialisation	Zero
Activation function	**ReLU**



(a) Change of loss function over epoch (b) Comparison with true function

Figure 4: Results of the second experiment.

non-linearity of activation functions. (He et al., 2015) Using Kaiming normal initialisation, the initialisation scheme is given as

$$W \sim \mathcal{N}\left(0, \sqrt{\frac{2}{n_{\text{in}}}}\right) \tag{19}$$

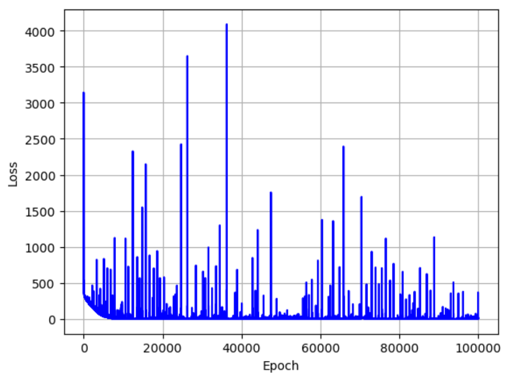
where n_{in} is the size of the previous layer.

Table 4 shows the hyperparameter settings for the third experiment.

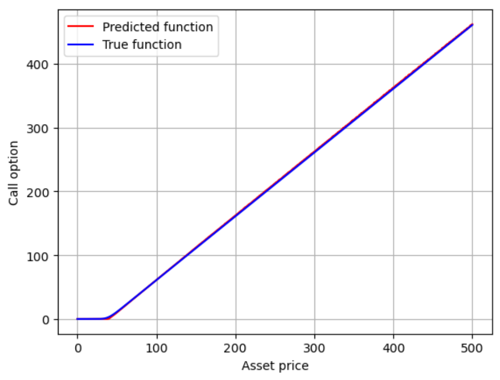
Table 4: Hyperparameter settings for third experiment

HYPERPARAMETER	**VALUE USED**
Learning Rate (η)	5.0×10^{-5}
Epochs	100,000
Number of sampled collocation points (N_f)	5000
Number of sample expiration points (N_e)	5000
Number of sample boundary points (N_b)	5000
Weight initialisation	**Kaiming Normal**
Activation function	**ReLU**

However, initialising weights using Kaiming initialisation scheme did not seem to improve the training performance compared to zero initialisation. Figure 5a shows the results of the third experiment, and figure 5b shows the comparison between the predicted function and the true function, calculated at $t = 0.5$. As shown in the graph, training loss experienced much more turbulence when compared to that of zero initialisation.



(a) Change of loss function over epoch



(b) Comparison with true function

Figure 5: Results of the third experiment.

Xavier initialisation is yet another famous choice of weight initialisation scheme for neural networks. (Glorot & Bengio, 2010) Xavier normal initialisation scheme initialises the weights at each layer to

$$W \sim \mathcal{N}\left(0, \sqrt{\frac{2}{n_{\text{in}} + n_{\text{out}}}}\right) \tag{20}$$

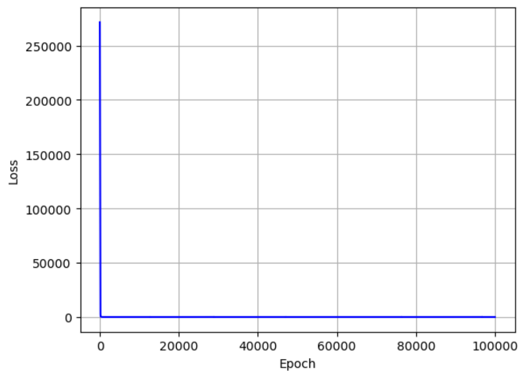
where n_{in} is the size of the previous layer, and n_{out} is the size of the next layer. Table 5 shows the hyperparameter settings for the fourth and final experiment.

Table 5: Hyperparameter settings for final experiment

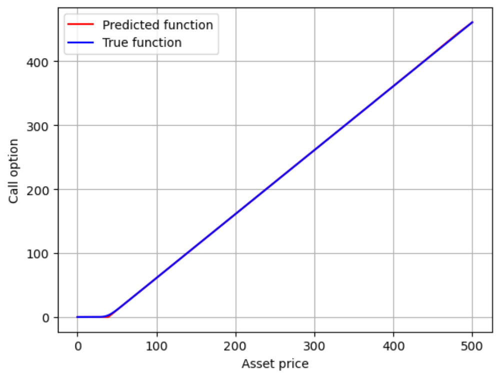
HYPERPARAMETER	**VALUE USED**
Learning Rate (η)	5.0×10^{-5}
Epochs	100,000
Number of sampled collocation points (N_f)	5000
Number of sample expiration points (N_e)	5000
Number of sample boundary points (N_b)	5000
Weight initialisation	**Xavier Normal**
Activation function	**ReLU**

Figure 6a shows the results of the final experiment, and figure 6b shows the comparison between the predicted function and the true function, calculated at $t = 0.5$.

As can be seen from the figure 6, Xavier normal initialisation scheme was able to approximate the solution function with much higher accuracy than the previous methods. Moreover, especially when compared to Kaiming normal initialisation scheme, Xavier normal initialisation scheme tend to train with much higher stability. The following figures 7a and 7b shows the 3D plot of our predicted function of the final experiment and the true function.

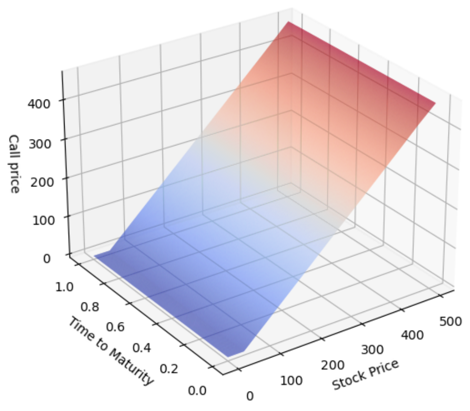


(a) Change of loss function over epoch

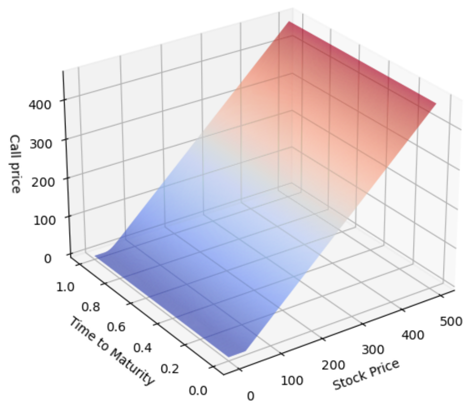


(b) Comparison with true function

Figure 6: Results of the final experiment.



(a) Predicted function



(b) True function

Figure 7: 3D plots of the functions.

5 Conclusion

In this project, I presented how Black-Scholes equation can be approximated via physics informed neural networks, and conducted several experiments under various hyperparameter circumstances. The model is trained with manually generated dataset and is optimised with Adam. The results showed that PINNs were able to approximate the solution function of Black–Scholes equation with extremely high accuracy.

There are several improvements applicable to vanilla PINNs. For instance, Sirignano & Spiliopoulos (2018) suggests a novel architecture for PINNs, known as ‘Deep Galerkin Method (DGM)’, that mimics the architecture of the well-known long short-term memory networks (LSTMs) (Hochreiter & Schmidhuber, 1997). Utilising the method enables the model to accurately solve partial differential equations in up to 200 dimensions.

Furthermore, this project can be easily extended to consider various different option markets. For instance, by modifying the loss function, it is possible to approximate the solution functions for Black-Scholes model with nonzero dividends. Another possible situation is to approximate the solution functions for Black-Scholes model with options styles other than European, such as American style or Asian style. In any cases, it is believed that the implemented PINN model in this project can be extended to solve each situation with ease.

ACKNOWLEDGMENTS

The baseline code used for constructing the overall network structure is retrieved from `https://github.com/nanditadoloi/PINN/`.

The code snippet for computing the analytic solution for Black-Scholes equation is retrieved from `https://aaronschlegel.me/black-scholes-formula-python.html/`.

The code snippet for generating 3D plots of the solution function of Black-Scholes equation is retrieved from `https://sigmaquant.wordpress.com/2016/06/18/interpreting-the-black-scholes-model/`.

I would like to express my sincerest gratitude.

REFERENCES

Amirhossein Arzani, Jian-Xun Wang, and Roshan M D'Souza. Uncovering near-wall blood flow from sparse data with physics-informed neural networks. *Physics of Fluids*, 33(7):071905, 2021.

Fischer Black. The pricing of commodity contracts. *Journal of financial economics*, 3(1-2):167–179, 1976.

Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of political economy*, 81(3):637–654, 1973.

Richard A Brealey, Stewart C Myers, Franklin Allen, and Pitabas Mohanty. *Principles of corporate finance*. Tata McGraw-Hill Education, 2012.

Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Ehsan Haghighat, Maziar Raissi, Adrian Moure, Hector Gomez, and Ruben Juanes. A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 379:113741, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Desmond J Higham. Black-scholes for scientific computing students. *Computing in Science & Engineering*, 6(6):72–79, 2004.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

Siddharth Krishna Kumar. On weight initialization in deep neural networks. *arXiv preprint arXiv:1704.08863*, 2017.

Stefano Markidis. Physics-informed deep-learning for scientific computing. *arXiv preprint arXiv:2103.09655*, 2021.

Bernt Oksendal. *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media, 2013.

Maziar Raissi and George Em Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *Journal of Computational Physics*, 357:125–141, 2018.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. 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, 2019.

Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.