Article

Three-Layer Artificial Neural Network for Pricing Multi-Asset European Option

Zhiqiang Zhou [1], Hongying Wu [2,*], Yuezhang Li [1], Caijuan Kang [1] and You Wu [1]

[1] School of Economics and Management, Xiangnan University, Chenzhou 423000, China; zq.zhou@xnu.edu.cn (Z.Z.); nicolelyz@sina.com (Y.L.); kcj622@xnu.edu.cn (C.K.); wuyou1996@xnu.edu.cn (Y.W.)
[2] School of Mathematics and Information Science, Xiangnan University, Chenzhou 423000, China
[*] Correspondence: wuhongying@xnu.edu.cn

Abstract: This paper studies an artificial neural network (ANN) for multi-asset European options. Firstly, a simple three-layer ANN-3 is established with undetermined weights and bias. Secondly, the time–space discrete PDE of the multi-asset option is given and the corresponding discrete data are fed into the ANN-3. Then, using least squares error as the objective function, the weights and bias of ANN-3 are trained well. Numerical examples are carried out to confirm the stability, accuracy and efficiency. Experiments show the ANN's relative error is about 0.8%. This method can be extended into multi-layer ANN-$q(q > 3)$ and extended into American options.

Keywords: multi-asset option; European option; high-dimensional PDE; artificial neural network; three layers

MSC: 65M99; 68W25; 65D12

Citation: Zhou, Z.; Wu, H.; Li, Y.; Kang, C.; Wu, Y. Three-Layer Artificial Neural Network for Pricing Multi-Asset European Option. *Mathematics* **2024**, *12*, 2770. https://doi.org/10.3390/math12172770

Academic Editor: Andrea Scozzari

Received: 8 August 2024
Revised: 31 August 2024
Accepted: 5 September 2024
Published: 7 September 2024

1. Introduction

In the last 30 years, there have been several methods for option pricing. For example, the radial basis function (RBF) method for options with a few assets [1–11]; analytical and semi-analytical methods for simple options [12,13]; finite difference method (FDM) for lower-dimensional option PDEs [14–18]; Monte Carlo simulation for complex options [19–21]; and Laplace transform or Mellin transform scheme [16,22–25] for classical options. Wu et al. discuss option pricing by the willow tree (WT) method for generalized hyperbolic Levý processes [26]. However, for RBF and FD methods, only lower-dimensional ($d \leq 2$) problems can be solved easily, and for the analytical or semi-analytical method, only some simple or classical options can be determined. In addition, although the Monte Carlo (MC) algorithm and the WT method are very simple and they have a wide application background, the MC method has low computational efficiency (consumes a lot of CPU time) and the WT method is of low convergence order, which limits their application.

In recent years, some of the literature starts to consider artificial neural networks (ANN) to solve option problems. Anderson and Ulrych discuss an accelerated American option pricing with deep neural networks [27]. Caverhill et al. give a neural network approach for option pricing and hedging [28]. Gan and Liu discuss option pricing based on the residual neural network [29]. Glau et al. discuss neural network expression rates and applications of the deep parametric PDE method in counter-party credit risk [30]. He and Guan study the parameter estimation method of the option pricing model based on the convolutional neural network in high-frequency financial trading [31]. Kapllani and Teng consider deep learning algorithms for solving high-dimensional nonlinear backward stochastic differential equations [32]. Lee and Son propose predicting arbitrage-free American option prices using an artificial neural network with pseudo inputs [33]. Mary and Salchenberger give a neural network model for estimating option prices [34]. Shvimer et al. discuss pricing options with a new hybrid neural network model [35]. Teng et al. discuss the combination of the neural network and classical volatility prediction model [36]. Tung et al. use a self-organizing neural-fuzzy semantic network for financial volatility trading [37]. Umeorah, Mashele and Agbaeze give barrier option pricing with neural networks [38]. Wang proposes a neural network forecasting model for the stock index option price [39]. However, none of the above literature is a neural network algorithm that is based on partial differential equations.

Unlike applying ANN to simple fitting problems, option pricing does not have natural samples, and we only know that the option satisfies a specific partial differential equation (PDE). To apply the ANN to option pricing, a key procedure is to discretize PDE and therefore make it a minimum optimization problem. To the best of our knowledge, this is the first time that ANN has been considered for option pricing from a multi-dimensional PDE.

In this paper, we propose an ANN to price the multi-dimensional option. The PDEs governed by the multi-asset option are discretized first, then a three-layer artificial neural network (ANN-3) is established. By minimizing the target error, all parameters (weights and bias) in ANN are trained well, then the neural network is applied. ANN-3 is easy to implement and has a powerful ability for multi-asset option pricing. Additionally, we propose some skills to accelerate the ANN training process.

The rest of this paper is formed as follows. In Section 2, the PDE governed by multi-asset option valuing is stated. In Section 3, a computational frame of the artificial neural network is proposed. We find out the computational formulas of derivatives for ANN weights and bias, and then we give the update formulas for the weights and bias. Section 4 gives some numerical examples to confirm the efficiency, accuracy and convergence of ANN. In Section 5, we give some conclusions and the future work. In Appendix A, some appendices are given, including algorithms and computational results.

2. PDE of Multi-Asset Option Values

Let $\boldsymbol{S} = (S_1(t), S_2(t), \dots, S_d(t))'$, with $(\cdot)'$ denoting the transposition of $(\cdot)$, be d-asset prices at time t. Let $x_I(t) = \log S_I(t), I = 1, 2, \dots, d$ be modeled by Brownian motion together with a drift term under no-arbitrage assumption, i.e., $x_I(t)$ are controlled by stochastic differential equations (SDEs),

$$\mathrm{d}x_I(t) = (r - q_I)\mathrm{d}t + \sigma_I \mathrm{d}W_t^{(I)}, \quad I = 1, 2, \dots, d. \tag{1}$$

In SDEs (1), r represents the risk-free interest, q_I represents the dividend yield, and σ_I is the volatility of asset I. $W_t^{(I)}, I = 1, 2, \dots, d$, are d standard Brownian motions, and it is assumed that

$$\begin{cases} \mathbb{E}\left(\mathrm{d}W_t^{(I)}\right) = 0, \quad Var\left(\mathrm{d}W_t^{(I)}\right) = \mathrm{d}t, \quad I = 1, 2, \dots, d, \\ Cov\left(\mathrm{d}W_t^{(I)}, \mathrm{d}W_t^{(J)}\right) = \rho_{IJ}\mathrm{d}t, \quad I, J = 1, 2, \dots, d, I \neq J, \end{cases} \tag{2}$$

where ρ_{IJ} is the coefficients of association between $\mathrm{d}W_t^{(I)}$ and $\mathrm{d}W_t^{(J)}$.

Let $u(\tau, \boldsymbol{x})$ be the option with $\tau = T - t$ and d log underlying $\boldsymbol{x} = (x_1(t), x_2(t), \dots, x_d(t))' \in \Omega = (-\infty, +\infty)^d$. Using the risk-free hedging theorem (see [40]), one can obtain the multi-dimensional PDE, named Black–Scholes equation, with the d-asset option value as

$$\frac{\partial u}{\partial \tau}(\tau, \boldsymbol{x}) = \mathcal{A}u(\tau, \boldsymbol{x}), \tau \in (0, T), \boldsymbol{x} \in \Omega, \tag{3}$$

where the linear differential operator $\mathcal{A}$ is defined as

$$\mathcal{A} := \sum_{I,J=1}^{d} a_{IJ} \frac{\partial^2}{\partial x_I \partial x_J} + \sum_{I=1}^{d} b_I \frac{\partial}{\partial x_I} - r, \tag{4}$$

with coefficients $a_{IJ} := \frac{1}{2}\rho_{IJ}\sigma_I\sigma_J$ and $b_I := r - q_I - \frac{1}{2}a_{II}$.

For the European option, $u(\tau, \boldsymbol{x})$ has initial values (also known as terminal conditions, or payoff function, for time $\tau = 0$)

$$u(0, \boldsymbol{x}) = \Pi(\boldsymbol{x}). \tag{5}$$

System (3)–(5) can be written as

$$\begin{cases} \frac{\partial u}{\partial \tau}(\tau, \boldsymbol{x}) - \mathcal{A}u(\tau, \boldsymbol{x}) = 0, \quad \tau \in (0, T], \ \boldsymbol{x} \in \Omega, \\ u(0, \boldsymbol{x}) = \Pi(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega. \end{cases} \tag{6}$$

For put options and geometric mean payoff function $\Pi(\boldsymbol{x})$, the initial values of u are taken as

$$u(0, \boldsymbol{x}) = \Pi(\boldsymbol{x}) = max\left(0, K - e^{\sum_{I=1}^{d} \alpha_I x_I}\right), \quad 0 < \alpha_I < 1, \ \sum_{I=1}^{d} \alpha_I = 1, \ \forall \boldsymbol{x} \in \Omega, \tag{7}$$

with strike price K. Ina general case, system (6) has no analytical solution, and a certain numerical method is needed to find the option value $u(\tau, \boldsymbol{x})$ in domain $(0, T] \times \Omega$.

With geometric mean payoff function $\Pi(\boldsymbol{x})$ defined by (7), the put option governed by system (6) has the analytical solution (see [40]),

$$u(\tau, \boldsymbol{x}) = Ke^{-r\tau}\Phi(-\widehat{d}_2) - e^{\sum_{I=1}^{d} \alpha_I x_I}\Phi(-\widehat{d}_1), \tag{8}$$

where $\Phi(\cdot)$ is the cumulative normal distribution function and

$$\widehat{d}_1 = \frac{1}{\widehat{\sigma}\sqrt{\tau}}\left[\sum_{I=1}^{d} \alpha_I x_I - \ln K + (r - \widehat{q} + \widehat{\sigma}^2/2)\tau\right], \quad \widehat{d}_2 = \widehat{d}_1 - \widehat{\sigma}\sqrt{\tau}, \tag{9}$$

with

$$\widehat{\sigma}^2 = \sum_{I,J=1}^{d} a_{IJ}\alpha_I\alpha_J, \quad \widehat{q} = \sum_{I=1}^{d} \alpha_I(q_I + a_{II}/2) - \widehat{\sigma}^2/2. \tag{10}$$

However, for the put option with arithmetic mean payoff function,

$$\Pi(\boldsymbol{x}) = max\left(0, K - \sum_{I=1}^{d} \alpha_I e^{x_I}\right), \quad 0 < \alpha_I < 1, \sum_{I}^{d} \alpha_I = 1, \forall \boldsymbol{x} \in \Omega, \tag{11}$$

the option $u(\tau, \boldsymbol{x})$ has no analytical solution. So, the numerical scheme is not a feasible method to price the multi-asset option governed by (6). The application of artificial neural networks to option pricing is discussed in the next section.

3. Computational Frame of ANN

An artificial neural network (ANN) is defined as in Figure 1. In this figure, the ANN-3 includes an input layer, one hidden layer and an output layer. For example, the neural units in the three layers are $n_1 = 3, n_2 = 4, n_3 = 1$, and we see in Section 4 that this structure is powerful for some examples.

The evolution of the 3-layer artificial neural network is represented as matrix expressions,

$$\begin{cases} \boldsymbol{O}^{(p)} = \boldsymbol{\omega}^{(p-1)}\boldsymbol{U}^{(p-1)} + \boldsymbol{\theta}^{(p)}, p = 2, 3, \\ \boldsymbol{U}^{(p)} = f_p(\boldsymbol{O}^{(p)}), p = 2, 3, \end{cases} \tag{12}$$

where $\boldsymbol{O}^{(p)}$ is the input data in the layer p, $\boldsymbol{U}^{(p)}$ is the output in the layer p, and f_p is the transform function(or activation function). The algebraic expression is given by

$$\begin{cases} O_j^{(p)} = \sum_{i=1}^{n_{p-1}} \omega_{ji}^{(p-1)} U_i^{(p-1)} + \theta_j^{(p)}, p = 2,3; j = 1,2,\ldots,n_p, \\ U_j^{(p)} = f_p(O_j^{(p)}), p = 2,3; j = 1,2\ldots,n_p, \end{cases} \tag{13}$$

where $U^{(1)}$ is a $(d+1)$ input data column vector $(x_1, x_2, \ldots, x_d, \tau)'$m and $f_p(\cdot)(p = 2,3)$ are the transform functions, i.e.,

$$f_2(x) = \frac{1}{1+e^{-x}} (\text{ sigmoid function }), \quad f_3(x) = x(\text{ linear function }). \tag{14}$$

We see the derivatives of transform functions $f_p(\cdot)$ are

$$\frac{\partial f_2(x)}{\partial x} = f_2(x)(1 - f_2(x)), \quad \frac{\partial f_3(x)}{\partial x} = 1. \tag{15}$$

The weights and bias in ANN are defined as

$$\boldsymbol{\omega}^{(2)} = \begin{bmatrix} \omega_{11}^{(2)} & \omega_{12}^{(2)} & \ldots & \omega_{1n_2}^{(2)} \end{bmatrix}, \boldsymbol{\omega}^{(1)} = \begin{bmatrix} \omega_{11}^{(1)} & \omega_{12}^{(1)} & \ldots & \omega_{1n_1}^{(1)} \\ \omega_{21}^{(1)} & \omega_{22}^{(1)} & \ldots & \omega_{2n_1}^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ \omega_{n_2 1}^{(1)} & \omega_{n_2 2}^{(1)} & \ldots & \omega_{n_2 n_1}^{(1)} \end{bmatrix}, \boldsymbol{\theta}^{(p)} = \begin{bmatrix} \theta_1^{(p)} \\ \theta_2^{(p)} \\ \vdots \\ \theta_{n_p}^{(p)} \end{bmatrix} \tag{16}$$

for $p = 2,3$. We call (12) or (13) an ANN-3 artificial neural network. Using the ANN-3 network has the following merits:

(1) It is relatively simple to compute the partial derivatives, $\Delta\boldsymbol{\omega}^{(i)}$ and $\Delta\boldsymbol{\theta}^{(j)}$, of $\boldsymbol{\omega}^{(i)}$ for $i = 1,2$ and $\boldsymbol{\theta}^{(j)}$ for $j = 2,3$ in ANN-3. Then we can easily update the parameters $\boldsymbol{\omega}^{(i)} = \boldsymbol{\omega}^{(i)} - \eta\Delta\boldsymbol{\omega}^{(i)}$ and $\boldsymbol{\theta}^{(j)} = \boldsymbol{\theta}^{(j)} - \eta\Delta\boldsymbol{\theta}^{(j)}$ with learning rate η.

(2) For deep learning network ANN-$q, q > 3$, it becomes very complicated to compute the partial derivatives of $\boldsymbol{\omega}^{(i)}$ and $\boldsymbol{\theta}^{(j)}$. So, we use ANN-$q$ for $q = 3$.

(3) For option pricing, we do not need deep learning networks. ANN-$q(q > 3)$ and the shallow artificial neural networks (ANN-3) are enough to solve the problem.

The output of the network is denoted by

$$U^{(3)} := U_1^{(3)}(\boldsymbol{X}), \quad \boldsymbol{U}^{(2)} = \left\{U_1^{(2)}(\boldsymbol{X}), U_2^{(2)}(\boldsymbol{X}), \ldots, U_{n_2}^{(2)}(\boldsymbol{X})\right\}, \quad U^{(1)} = \boldsymbol{X}, \tag{17}$$

with input data $\boldsymbol{X} = [x_1, x_2, \ldots, x_d, \tau]'$. We denote by $U_1^{(3)}(\boldsymbol{X}) := net\left(\boldsymbol{\omega}^{(1)}, \boldsymbol{\omega}^{(2)}, \boldsymbol{\theta}^{(2)}, \boldsymbol{\theta}^{(3)}, \mathbf{X}\right)$ the network output with input data $\mathbf{X} = (x_1, x_2, \cdots, x_d, \tau)'$ under parameters $\boldsymbol{\omega}^{(1)}, \boldsymbol{\omega}^{(2)}, \boldsymbol{\theta}^{(2)}$ and $\boldsymbol{\theta}^{(3)}$. Furthermore, we denote by $U^{(p,k)} = U^{(p)}(\boldsymbol{X}^{(k)}), p = 1,2,3$ with N space and time input data

$$\boldsymbol{X}^{(k)} = \left[x_1^{(k)}, x_2^{(k)}, \ldots, x_d^{(k)}, \tau^{(k)}\right]^T, k = 1,2,\ldots,N. \tag{18}$$

The purpose of our ANN structure is to determine the optimal weights $\boldsymbol{\omega}^{(1)}, \boldsymbol{\omega}^{(2)}$ and bias $\boldsymbol{\theta}^{(2)}, \boldsymbol{\theta}^{(3)}$. For this purpose, we use an iteration algorithm, i.e., we modify the values of the parameters

$$\left(\boldsymbol{\omega}^{(1)}, \boldsymbol{\omega}^{(2)}, \boldsymbol{\theta}^{(2)}, \boldsymbol{\theta}^{(3)}\right)$$

until some objective function MSE is less than the pre-specified error ε.

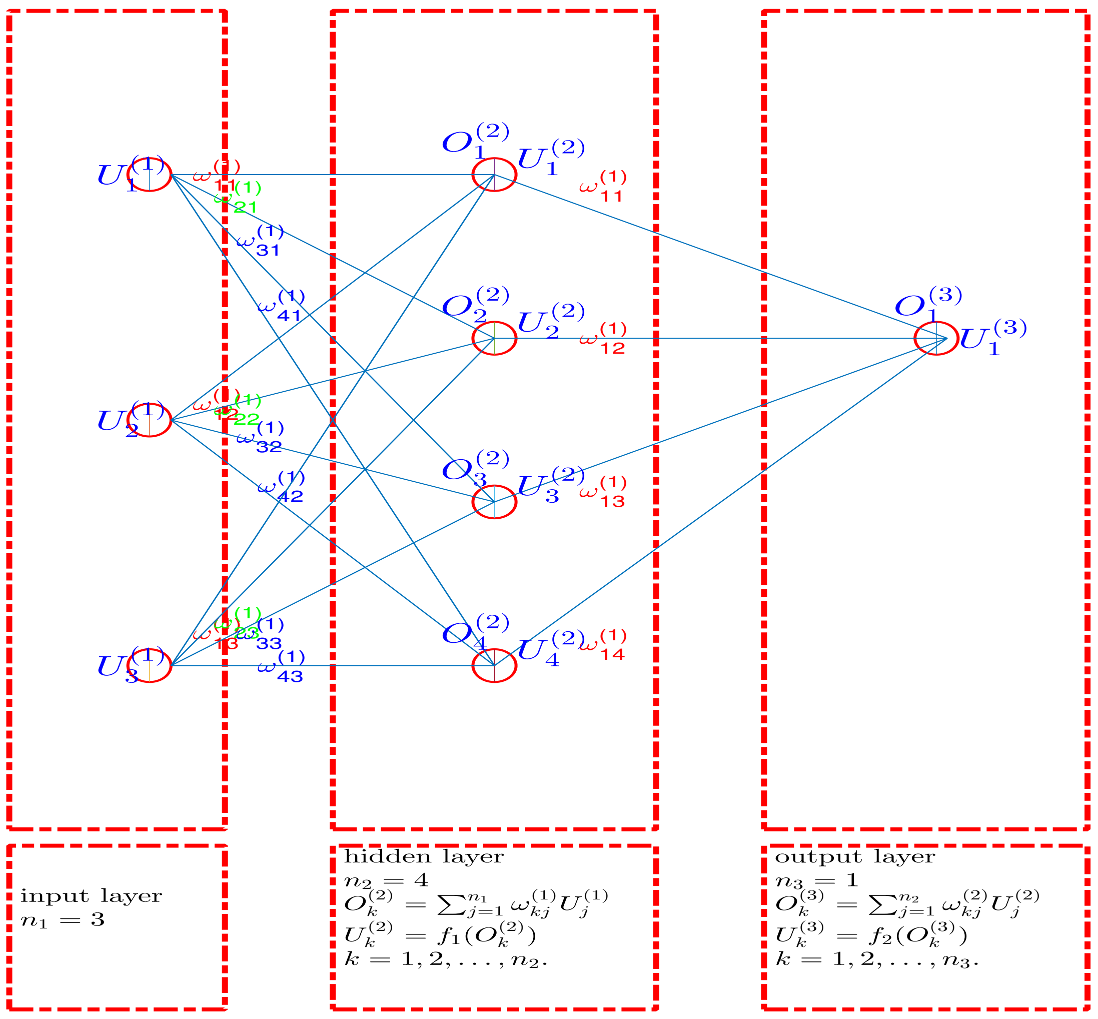
$U_1^{(1)}$
$U_2^{(1)}$
$U_3^{(1)}$
$O_1^{(2)}$ $U_1^{(2)}$
$O_2^{(2)}$ $U_2^{(2)}$
$O_3^{(2)}$ $U_3^{(2)}$
$O_4^{(2)}$ $U_4^{(2)}$
$O_1^{(3)}$ $U_1^{(3)}$
$\omega_{11}^{(1)}$ $\omega_{21}^{(1)}$ $\omega_{31}^{(1)}$ $\omega_{41}^{(1)}$
$\omega_{12}^{(1)}$ $\omega_{22}^{(1)}$ $\omega_{32}^{(1)}$ $\omega_{42}^{(1)}$
$\omega_{13}^{(1)}$ $\omega_{23}^{(1)}$ $\omega_{33}^{(1)}$ $\omega_{43}^{(1)}$
$\omega_{11}^{(1)}$ $\omega_{12}^{(1)}$ $\omega_{13}^{(1)}$ $\omega_{14}^{(1)}$
input layer $n_1 = 3$
hidden layer $n_2 = 4$ $O_k^{(2)} = \sum_{j=1}^{n_1} \omega_{kj}^{(1)} U_j^{(1)}$ $U_k^{(2)} = f_1(O_k^{(2)})$ $k = 1, 2, \ldots, n_2.$
output layer $n_3 = 1$ $O_k^{(3)} = \sum_{j=1}^{n_2} \omega_{kj}^{(2)} U_j^{(2)}$ $U_k^{(3)} = f_2(O_k^{(3)})$ $k = 1, 2, \ldots, n_3.$

Figure 1. Graphical depiction of the artificial neural network ANN-3 with an input layer, one hidden layer and a output layer.

For the convenience, we list some of the symbols in this paper in Table 1.

Table 1. Some symbols for ANN.

Name	Meaning
N'	The number of data that are the initial conditions
N	The number of input data for the training of ANN
$N - N'$	The input number of data for discrete PDE
$net(\boldsymbol{\omega}^{(1)}, \boldsymbol{\omega}^{(2)}, \boldsymbol{\theta}^{(2)}, \boldsymbol{\theta}^{(3)})$	ANN with weights $(\boldsymbol{\omega}^{(1)}, \boldsymbol{\omega}^{(2)})$ and bias $(\boldsymbol{\theta}^{(2)}, \boldsymbol{\theta}^{(3)})$
$U^3(\boldsymbol{X}) = net\left(\boldsymbol{\omega}^{(1)}, \boldsymbol{\omega}^{(2)}, \boldsymbol{\theta}^{(2)}, \boldsymbol{\theta}^{(3)}, \mathbf{X}\right)$	ANN output with input data $\mathbf{X} = (x_1, x_2, \ldots, x_d, \tau)'$
η	Learning rate of ANN
$\boldsymbol{O}_k^{(p)}$	Input data at the p layer $(p = 2, q)$
$\boldsymbol{U}_k^{(p)}$	Output data at the p layer $(p = 1, 2, q)$
$ERR^{(payoff)}$	Error for initial values
$ERR^{(PDE)}$	Error for discrete PDE
$\Delta\boldsymbol{\theta}^{(payoff,\cdot)}$, $\Delta\boldsymbol{\theta}^{(PDE,\cdot)}$	Partial derivatives for $\boldsymbol{\theta}$
$\Delta\boldsymbol{\omega}^{(payoff,\cdot)}$, $\Delta\boldsymbol{\omega}^{(PDE,\cdot)}$	Partial derivatives for $\boldsymbol{\omega}$
D	Partial derivatives for $\boldsymbol{x}$ with $D = D_\tau, D = D_I$ and $D = D_{IJ}$

In the next paragraphs, we give the definition of objective function for the network, the mean square error (MSE),

$$MSE = MSE^{(payoff)} + MSE^{(PDE)}, \tag{19}$$

with $MSE^{(payoff)}$ being the error at points $\boldsymbol{X}^{(k)} = \left(x_1^{(k)}, \dots, x_d^{(k)}, \tau_1\right)$ with $k = 1, 2, \dots, N'$ and $\tau_1 = 0$, and $MSE^{(PDE)}$ being the error of the PDE (6) with $\boldsymbol{X}^{(k)} = \left(x_1^{(k)}, \dots, x_d^{(k)}, \tau^{(k)}\right)$ and $\tau^{(k)}$ being one of the values

$$\tau^{(k)} = \Delta\tau, \dots, \tau_j, \dots, \tau_M = T. \tag{20}$$

Firstly, we consider the $MSE^{(payoff)}$ at $\tau_1 = 0$.

For the initial conditions, we have $U_1^{(3)}(\boldsymbol{X}^{(k)}) = \Pi(\boldsymbol{x}^{(k)}) := y^{(k)}, k = 1, 2, \dots, N'$. Then the $MSE^{(payoff)}$ of ANN at $\tau_1 = 0$ is defined by

$$MSE^{(payoff)} = \frac{1}{2N'} \sum_{k=1}^{N'} \left(U_1^{(3,k)} - y^{(k)}\right)^2. \tag{21}$$

The partial derivatives of $MSE^{(payoff)}$ with respect to $\boldsymbol{\omega}^{(2)} = \left[\omega_{11}^{(2)}, \dots, \omega_{1\ell}^{(12)}, \dots, \omega_{1n_2}^{(2)}\right]$ are

$$\frac{\partial MSE^{(payoff)}}{\partial \omega_{1\ell}^{(2)}} = \frac{\partial MSE^{(payoff)}}{\partial U_1^{(3,k)}} \frac{\partial U_1^{(3,k)}}{\partial O_1^{(3,k)}} \frac{\partial O_1^{(3,k)}}{\partial \omega_{1\ell}^{(2)}} = \frac{1}{N'} \sum_{k=1}^{N'} \left(U_1^{(3,k)} - y^{(k)}\right) U_\ell^{(2,k)}, \tag{22}$$

for $\ell = 1, 2, \dots, n_2$, and the partial derivative with respect to $\theta_1^{(3)}$ is

$$\frac{\partial MSE^{(payoff)}}{\partial \theta_1^{(3)}} = \frac{\partial MSE^{(payoff)}}{\partial U_1^{(3,k)}} \frac{\partial U_1^{(3,k)}}{\partial O_1^{(3,k)}} \frac{\partial O_1^{(3,k)}}{\partial \theta_1^{(3)}} = \frac{1}{N'} \sum_{k=1}^{N'} \left(U_1^{(3,k)} - y^{(k)}\right). \tag{23}$$

The partial derivatives of $MSE^{(payoff)}$ with respect to $\omega_{j\ell}^{(1)}$ are

$$\begin{aligned} \frac{\partial MSE^{(payoff)}}{\partial \omega_{j\ell}^{(1)}} &= \frac{\partial MSE^{(payoff)}}{\partial U_1^{(3,k)}} \frac{\partial U_1^{(3,k)}}{\partial O_1^{(3,k)}} \frac{\partial O_1^{(3,k)}}{\partial U_j^{(2,k)}} \frac{\partial U_j^{(2,k)}}{\partial O_j^{(2,k)}} \frac{\partial O_j^{(2,k)}}{\partial \omega_{j\ell}^{(1)}} \\ &= \frac{1}{N'} \sum_{k=1}^{N'} \left(U_1^{(3,k)} - y^{(k)}\right) \omega_{1j}^{(2)} U_j^{(2,k)} \left(1 - U_j^{(2,k)}\right) U_\ell^{(1,k)}, \end{aligned} \tag{24}$$

for $j = 1, 2, \dots, n_2$ and $\ell = 1, 2, \dots, n_1$. The partial derivatives of $MSE^{(payoff)}$ with respect to $\theta_j^{(2)}$ are

$$\begin{aligned} \frac{\partial MSE^{(payoff)}}{\partial \theta_j^{(2)}} &= \frac{\partial MSE^{(payoff)}}{\partial U_1^{(3,k)}} \frac{\partial U_1^{(3,k)}}{\partial O_1^{(3,k)}} \frac{\partial O_1^{(3,k)}}{\partial U_j^{(2,k)}} \frac{\partial U_j^{(2,k)}}{\partial O_j^{(2,k)}} \frac{\partial O_j^{(2,k)}}{\partial \theta_j^{(2)}} \\ &= \frac{1}{N'} \sum_{k=1}^{N'} \left(U_1^{(3,k)} - y^{(k)}\right) \omega_{1j}^{(2)} U_j^{(2,k)} \left(1 - U_j^{(2,k)}\right), \end{aligned} \tag{25}$$

for $j = 1, 2, \dots, n_2$. Let

$$\begin{cases} \Delta\boldsymbol{\omega}^{(payoff,2)} = \left[\dfrac{\partial MSE^{(payoff)}}{\partial \omega_{1j}^{(2)}} \right]_{1\times n_2}, \\ \Delta\boldsymbol{\omega}^{(payoff,1)} = \left[\dfrac{\partial MSE^{(payoff)}}{\partial \omega_{j\ell}^{(1)}} \right]_{n_2\times n_1} \\ \Delta\boldsymbol{\theta}^{(payoff,3)} = \dfrac{1}{N'} \sum_{k=1}^{N} \left(U^{(3,k)} - y^{(k)} \right), \\ \Delta\boldsymbol{\theta}^{(payoff,2)} = \left[\dfrac{\partial MSE^{(payoff)}}{\partial \theta_{j}^{(2)}} \right]_{n_2\times 1}, \end{cases} \tag{26}$$

with the definition of partial derivatives (22)–(25).

Secondly, we discuss the $MSE^{(PDE)}$ at discrete points $\boldsymbol{X}^{(k)}, k = N'+1, 2, \ldots, N$. The discrete error $MSE^{(PDE)}$ of PDE at $\boldsymbol{X}^{(k)}$ for $k = N'+1, \ldots, N$ is defined by

$$MSE^{(PDE)} := \frac{1}{2(N-N')} \sum_{k=N'+1}^{N} ERR_k^2, \tag{27}$$

where

$$\begin{aligned} ERR_k &:= \left[\frac{\partial U(\tau,\boldsymbol{x})}{\partial \tau} - \mathcal{A}U(\tau,\boldsymbol{x}) \right]_{\boldsymbol{X}=\boldsymbol{X}^{(k)}} \\ &\approx D_\tau U^{(3,k)} - \sum_{I=1}^{d}\sum_{J=1}^{d} a_{IJ} D_{IJ} U^{(3,k)} - \sum_{I=1}^{d} b_I D_I U^{(3,k)} + rU^{(3,k)}, \end{aligned} \tag{28}$$

for $k = N'+1, \ldots, N$. Here, differential operators $D_\tau := \frac{\partial}{\partial \tau}, D_I := \frac{\partial}{\partial x_I}$ and $D_{IJ} := \frac{\partial^2}{\partial x_I \partial x_J}$. The partial derivatives are defined as

$$\begin{cases} D_{IJ}U_1^{(3,k)} := \dfrac{\partial^2}{\partial x_I \partial x_J} U_1^{(3,k)} = \dfrac{\partial}{\partial x_J}\left(\dfrac{\partial}{\partial x_I} U_1^{(3,k)} \right), \\ D_I U_1^{(3,k)} := \dfrac{\partial}{\partial x_I} U_1^{(3,k)} = \dfrac{\partial U_1^{(3,k)}}{\partial O_1^{(3,k)}} \sum_{j=1}^{n_2} \dfrac{\partial O_1^{(3,k)}}{\partial U_j^{(2,k)}} \dfrac{\partial U_j^{(2,k)}}{\partial O_j^{(2,k)}} \dfrac{\partial O_j^{(2,k)}}{\partial U_I^{(1,k)}}, \\ D_\tau U_1^{(3,k)} := \dfrac{\partial}{\partial \tau} U_1^{(3,k)} = \dfrac{\partial U_1^{(3,k)}}{\partial O_1^{(3,k)}} \sum_{j=1}^{n_2} \dfrac{\partial O_1^{(3,k)}}{\partial U_j^{(2,k)}} \dfrac{\partial U_j^{(2,k)}}{\partial O_j^{(2,k)}} \dfrac{\partial O_j^{(2,k)}}{\partial U_{d+1}^{(1,k)}}, \end{cases} \tag{29}$$

for $k = N'+1, 2, \cdots, N$, which induces

$$\begin{cases} D_{IJ}U_1^{(3,k)} = \sum_{j=1}^{n_2} \omega_{1j}^{(2)} \omega_{jI}^{(1)} \omega_{jJ}^{(1)} U_j^{(2,k)} \left(1 - U_j^{(2,k)}\right)\left(1 - 2U_j^{(2,k)}\right), \\ D_I U_1^{(3,k)} = \sum_{j=1}^{n_2} \omega_{1j}^{(2)} \omega_{jI}^{(1)} U_j^{(2,k)} \left(1 - U_j^{(2,k)}\right), \\ D_\tau U_1^{(3,k)} = \sum_{j=1}^{n_2} \omega_{1j}^{(2)} \omega_{j,d+1}^{(1)} U_j^{(2,k)} \left(1 - U_j^{(2,k)}\right), \end{cases} \tag{30}$$

for $I, J = 1, 2, \ldots, d$.

Define the partial derivatives

$$\frac{\partial MSE^{(PDE)}}{\partial \omega_{j\ell}^{(m)}} = \frac{1}{2(N-N')} \sum_{k=N'+1}^{N} \frac{\partial ERR_k^2}{\partial \omega_{j\ell}^{(m)}}, \tag{31}$$

for $m = 2,1$, $k = N' + 1, \dots, N, j = 1,2,\dots,n_{m+1}$ and $\ell = 1,2,\dots,n_m$. Define the partial derivatives

$$\frac{\partial MSE^{(PDE)}}{\partial \theta_j^{(m)}} = \frac{1}{2(N-N')} \sum_{k=N'+1}^{N} \frac{\partial ERR_k^2}{\partial \theta_j^{(m)}}, \tag{32}$$

for $m = 3,2, j = 1,2,\dots,n_m$. The partial derivatives $\frac{1}{2}\frac{\partial ERR_k^2}{\partial \omega_{j\ell}^{(m)}}$ are found as

$$\begin{aligned} \frac{1}{2}\frac{\partial ERR_k^2}{\partial \omega_{j\ell}^{(m)}} &= ERR_k \frac{\partial ERR_k}{\partial \omega_{j\ell}^{(m)}} \\ = \quad & ERR_k \left\{ \frac{\partial\left(D_\tau U_1^{(3,k)}\right)}{\partial \omega_{j\ell}^{(m)}} - \sum_{I=1}^{d}\sum_{J=1}^{d} a_{IJ} \frac{\partial\left(D_{IJ} U_1^{(3,k)}\right)}{\partial \omega_{j\ell}^{(m)}} - \sum_{I=1}^{d} b_I \frac{\partial(D_I U_1^{(3,k)})}{\partial \omega_{j\ell}^{(m)}} + r \frac{\partial U_1^{(3,k)}}{\partial \omega_{j\ell}^{(m)}} \right\}, \end{aligned} \tag{33}$$

for $m = 1,2$, for $j = 1,2,\dots,n_{m+1}$ and $\ell = 1,2,\dots,n_m$. The partial derivatives $\frac{1}{2}\frac{\partial ERR_k^2}{\partial \theta_j^{(m)}}$ are found as

$$\begin{aligned} \frac{1}{2}\frac{\partial ERR_k^2}{\partial \theta_j^{(m)}} &= ERR_k \frac{\partial ERR_k}{\partial \theta_j^{(m)}} \\ = \quad & ERR_k \left\{ \frac{\partial D_\tau U_1^{(3,k)}}{\partial \theta_j^{(m)}} - \sum_{I=1}^{d}\sum_{J=1}^{d} a_{IJ} \frac{\partial\left(D_{IJ} U_1^{(3,k)}\right)}{\partial \theta_j^{(m)}} - \sum_{I=1}^{d} b_I \frac{\partial(D_I U_1^{(3,k)})}{\partial \theta_j^{(m)}} + r \frac{\partial U_1^{(3,k)}}{\partial \theta_j^{(m)}} \right\}, \end{aligned} \tag{34}$$

for $j = 1,2,\dots,n_m$ and $m = 2,3$.

In (33), the partial derivatives $\frac{\partial (DU_1)^{(3,k)}}{\partial \omega_{j\ell}^{(m)}}$ with $D = D_\tau, D = D_{IJ}$ or $D = D_I$ are defined as

$$\left\{ \begin{aligned} \frac{\partial (DU_1)^{(3,k)}}{\partial \omega_{1j}^{(2)}} &= (DU_1)_j^{(2,k)}, \\ \frac{\partial (DU_1)^{(3,k)}}{\partial \omega_{j\ell}^{(1)}} &= \omega_{1j}^{(2)} (DU_j)^{(2,k)} \left(1 - (DU_j)^{(2,k)}\right) (DU_\ell)^{(1,k)}, \end{aligned} \right. \tag{35}$$

for $j = 1,2,\cdots,n_2$ and $\ell = 1,2,\cdots,n_1$.

In (34), the partial derivatives $\frac{\partial (DU_1)^{(3,k)}}{\partial \theta_j^{(m)}}$ with $D = D_\tau, D = D_{IJ}$ or $D = D_I$ are defined as

$$\left\{ \begin{aligned} \frac{\partial (DU_1)^{(3,k)}}{\partial \theta_1^{(3)}} &= 1, \\ \frac{\partial (DU_1)^{(3,k)}}{\partial \theta_j^{(2)}} &= \omega_{1j}^{(2)} (DU_j)^{(2,k)} \left(1 - (DU_j)^{(2,k)}\right), \end{aligned} \right. \tag{36}$$

for $j = 1,2,\cdots,n_2$.

Define the increments of $\boldsymbol{\omega}^{(PDE,i)}$ and $\boldsymbol{\theta}^{(PDE,i)}$ as follows:

$$\begin{cases} \Delta\boldsymbol{\omega}^{(PDE,2)} = \dfrac{1}{2(N-N')}\displaystyle\sum_{k=N'+1}^{N}\left[\dfrac{\partial ERR_k^{(2)}}{\partial \omega_{1j}^{(2)}}\right]_{1\times n_2}, \\ \Delta\boldsymbol{\omega}^{(PDE,1)} = \dfrac{1}{2(N-N')}\displaystyle\sum_{k=N'+1}^{N}\left[\dfrac{\partial ERR_k^{(2)}}{\partial \omega_{ij}^{(1)}}\right]_{n_2\times n_1}, \\ \Delta\boldsymbol{\theta}^{(PDE,3)} = 1, \\ \Delta\boldsymbol{\theta}^{(PDE,2)} = \dfrac{1}{2(N-N')}\displaystyle\sum_{k=N'+1}^{N}\left[\dfrac{\partial ERR_k^{(2)}}{\partial \theta_{j}^{(2)}}\right]_{n_2\times 1}, \end{cases} \tag{37}$$

with definitions (31) and (32).

Then, we update the network's weights $\boldsymbol{\omega}^{(2)}, \boldsymbol{\omega}^{(1)}, \boldsymbol{\theta}^{(3)}$ and $\boldsymbol{\theta}^{(2)}$ as follows:

$$\begin{cases} \boldsymbol{\omega}^{(2)} = \boldsymbol{\omega}^{(2)} - \eta\left(\Delta\boldsymbol{\omega}^{(payoff,2)} + \Delta\boldsymbol{\omega}^{(PDE,2)}\right), \\ \boldsymbol{\omega}^{(1)} = \boldsymbol{\omega}^{(1)} - \eta\left(\Delta\boldsymbol{\omega}^{(payoff,1)} + \Delta\boldsymbol{\omega}^{(PDE,1)}\right), \\ \boldsymbol{\theta}^{(3)} = \boldsymbol{\theta}^{(3)} - \eta(\Delta\boldsymbol{\theta}^{(payoff,3)} + \Delta\boldsymbol{\theta}^{(PDE,3)}), \\ \boldsymbol{\theta}^{(2)} = \boldsymbol{\theta}^{(2)} - \eta(\Delta\boldsymbol{\theta}^{(payoff,2)} + \Delta\boldsymbol{\theta}^{(PDE,2)}), \end{cases} \tag{38}$$

where $\eta = constant$ is the learning rate and $\Delta\boldsymbol{\omega}^{(*,j)}$ and $\Delta\boldsymbol{\theta}^{(*,j)}$ are defined by (26) and (37).

Finally, define the total error as in (19) and

$$\begin{cases} MSE^{(payoff)} = \dfrac{1}{2N'}\displaystyle\sum_{k=1}^{N'}\left(U_1^{(3,k)} - y^{(k)}\right)^2, \text{ (from the initial constrains)}, \\ MSE^{(PDE)} = \dfrac{1}{2(N-N')}\displaystyle\sum_{k=N'+1}^{N}(ERR_k)^2, \text{ (from the discrete PDE)}, \end{cases} \tag{39}$$

with ERR_k being defined by (28). To minimize the objective function MSE, we have

$$\left\{\boldsymbol{\omega}^{(i)}, \boldsymbol{\theta}^{(j)}\right\}_{i=1,2;j=2,3} = \operatorname{argmin} MSE = \operatorname{argmin}\left[MSE^{(payoff)} + MSE^{(PDE)}\right]. \tag{40}$$

and we obtain the trained network $net\left(\boldsymbol{\omega}^{(1)}, \boldsymbol{\omega}^{(2)}, \boldsymbol{\theta}^{(2)}, \boldsymbol{\theta}^{(3)}, \{U^{(1,k)}\}_{k=1}^{N}\right)$ with input samples $\{U^{(1,k)}\}_{k=1}^{N}$ and optimal parameters $(\boldsymbol{\omega}^{(1)}, \boldsymbol{\omega}^{(2)}, \boldsymbol{\theta}^{(2)}, \boldsymbol{\theta}^{(3)})$ by iterating algorithm (38). Finally, we can compute

$$U_1^{(3)}(\mathbf{X}) = net\left(\boldsymbol{\omega}^{(1)}, \boldsymbol{\omega}^{(2)}, \boldsymbol{\theta}^{(2)}, \boldsymbol{\theta}^{(3)}, \mathbf{X}\right), \tag{41}$$

for any input data $\mathbf{X} = (x_1, x_2, \cdots, x_d, \tau)'$.

The Algorithm A1 in Appendix A.1 gives the detailed procedure of the artificial neural network for option pricing.

4. Numerical Examples

4.1. Parameter Setting and Selection of Learning Rate

In computational experiments, we set parameters for the multi-asset option as follows. The dimension of PDE is set as $d = 2, 3, 4$. The coefficients are set as $\rho_{ij} = 0.8 (i \neq j)$ and $\rho_{ii} = 1$ for $i, j = 1, 2, \ldots, d$. The dividend yields are set as $q_i = 0.002$, and risk-free interest is set as $r = 0.05$. The volatilities are set as $\sigma_i = 0.1$ for $i = 1, 2, \ldots, d$. Strike prices are set as

$K = 8, 10$. Asset ratios are set as $\alpha_i = 1/d$ for $i = 1, 2, \ldots, d$. The maturity date is taken as $T = 0.4$.

The discrete stock prices are taken as $x_i = \log(0.02) + \frac{i}{n}[\log(70) - \log(0.02)]$ with $i = 1, 2, \ldots, n$ and $n = 5$. The time to maturity τ is discretized as $\tau_j = j\Delta\tau$ with $j = 0, 1, \ldots, M-1$, $M = T/\Delta\tau$ and $\Delta\tau = 0.1$. The input data for network training are expressed by $\boldsymbol{X} = [x_1^{(k)}, x_2^{(k)}, \ldots, x_d^{(k)}, \tau^{(k)}]'$ with each $x_i^{(k)}$ being one of x_i and each $\tau^{(k)}$ being one of τ_j. So, the total number of training data is $n^d M$. The input data $\boldsymbol{X} = [x_1^{(k)}, x_2^{(k)}, \ldots, x_d^{(k)}, 0]', k = 1, 2, \ldots, N'$ are taken as the initial data with network output being the payoff functions $\Pi(\boldsymbol{X})$ (see (7) and (11)). The reminder data $\boldsymbol{X} = [x_1^{(k)}, x_2^{(k)}, \ldots, x_d^{(k)}, \tau^{(k)}]', k = N'+1, \ldots, N$ with $\tau^{(k)} \neq 0$ are taken as the input data corresponding to the discrete PDE.

The network layers are taken as $n_1 = d+1, n_2 = 5$ and $n_3 = 1$, and the training number is taken as $L = 1000$ in the ANN structure.

Throughout the training process, we use the following techniques:

(1) The learning rate η is set as $\eta = 1$ initially. When the objective function is not decreasing, we set $\eta = 0.95\eta$. Throughout the training process, we use a factor of 0.95 to reduce the learning rate when the error does not decrease, repeatedly.

(2) To speed the training process, at each iteration, we only use partial training data to update the weights $\boldsymbol{\omega}^{(i)}$ and the bias $\boldsymbol{\theta}^{(i)}$. Simply, at the i^{th} iteration, we use the data sequence labeled by $mod(i, N') : N' : N$ as the input data of ANN.

(3) The simulation result is $U^3(\boldsymbol{X}) = net\left(\boldsymbol{\omega}^{(1)}, \boldsymbol{\omega}^{(2)}, \boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \mathbf{X}\right)$ for any input data $\boldsymbol{X}$ (see expression (41)) contained within the envelope of $\{\boldsymbol{X}^{(k)}, k = 1, 2, \cdots, N\}$, with trained $\boldsymbol{\omega}^{(i)}$ and $\boldsymbol{\theta}^{(j)}$.

The Monte Carlo algorithm for arithmetic mean payoff function is described as in Algorithm A2 in Appendix A.1. For geometric mean payoff function, we modify the simulated option $P^{(k)} = max(0, K - \sum_{i=1}^{d} e^{\alpha_i x_{M-1,i}^{(k)}})$ at Step 3.

4.2. Numerical Results with Geometric Mean Payoff Function

With geometric mean payoff function (7), the multi-asset put option has analytical solution (8). We use some numerical examples to illustrate the errors between ANN numerical solutions and analytical options.

Tables A1–A3 list the ANN solutions (labeled with 'ANN') and the analytical solutions (labeled with 'Anal.') with $d = 2, 3, 4$, time to expiry date $\tau = 0.4$ and different strike prices of $K = 8, 10$. In Tables A1–A3, the fourth column and the fifth column are the absolute errors (labeled with 'ERR') and the relative errors (labeled with 'RE'), respectively. Figure A1 plots the analytical solutions and ANN solutions at different asset points with parameters: $T = 0.4, K = 8, 10$ and $d = 2$.

From Tables A1–A3, we see the absolute errors are about 0.05 and the relative errors are about 0.08% between ANN solutions and analytical solutions, which illustrate that our ANN scheme is efficient and accurate. Moreover, the whole results of calculation are quite stable, there is no unstable result.

The Figure A2 records the error evolution path with respect to training number L. We see from this figure, the errors are decreasing quickly as the iteration number L increases. As iteration number L increases, the Figure A3 records the learning rate path η. The learning rate η decreases with the increase in L from 1 to 0.1.

Tables A1–A3 and Figures A1–A3 show that our ANN-3 is efficient, accurate and computationally stable for option pricing for geometric mean payoff function.

Lastly, according to the results of numerical examples, we give a conjecture about the errors between ANN solutions and analytical solutions of multi-asset options, which is the following conjecture.

Conjecture 1. *Assume $U_1^{(3)}(\boldsymbol{X}) = net(\boldsymbol{\omega}^{(1)}, \boldsymbol{\omega}^{(2)}, \boldsymbol{\theta}^{(2)}, \boldsymbol{\theta}^{(3)}, \boldsymbol{X})$ is the network solution and $u(\boldsymbol{X})$ is the exact solution of multi-asset option. Then, the error between $U_1^{(3)}(\boldsymbol{X})$ and $u(\boldsymbol{X})$ can be estimated by*

$$|U_1^{(3)}(\boldsymbol{X}) - u(\boldsymbol{X})| \leq O\left(\frac{C}{n_2}\right), \tag{42}$$

for any asset prices and any remainder time $\boldsymbol{X} = [\boldsymbol{x}', \tau]$, where n_2 is the number of neurons in ANN hidden layer, and C is a certain constant.

Figure A4 plots the error vs. the number n_2 of ANN, from which we see the ANN error is about $O(\frac{C}{n_2})$ with C being a positive constant, which confirms Conjecture 1.

4.3. Numerical Results with Arithmetic Mean Payoff Function

In this subsection, we list some results of the put options with arithmetic mean payoff function $\Pi(\boldsymbol{x}) = max\left(0, \sum_{i=1}^{d} \alpha_i e^{x_i}\right)$. This type of option has no analytical solution. We give the Monte Carlo simulation (the procedure can be seen in Algorithm A2) as the compared results.

Tables A4–A6 list the compared results of ANN solutions (labeled with 'ANN') and Monte Carlo simulation (labeled with 'MC'). From these tables, we see the absolute errors (labeled with 'ERR') are less than 2×10^{-2} and the relative errors (labeled with 'RE') are less than 0.8%, which illustrate that our ANN algorithm is effective and accurate. Figure A5 plots the compared solutions obtained by the ANN method and the Monte Carlo simulation with $d = 2, T = 0.4$ and $K = 8$.

Figure A6 records the errors with respect to iteration number L. From this figure, we see the errors are decreasing quickly as the iteration number L increases. Figure A7 records the learning rate path as the iteration number L increases. From this figure, we see that as the iteration number progresses, the learning rate decreases from 1 to 0.01.

Tables A4–A6 and Figures A5–A7 illustrate that our ANN-3 is efficient, accurate and computationally stable for option pricing for arithmetic mean payoff function.

5. Conclusions

This paper gives an artificial neural network (ANN) with three layers to price the European multi-asset options with the input data $\boldsymbol{X}$ from discrete PDE. By setting the numbers (n_1, n_2, n_3) of the ANN structure, the multi-asset options are simulated correctly. The key technology is obtaining the discrete formula of PDE and then computing the partial derivatives of the objective function with respect to weights $\omega_{ij}^{(k)}$ and bias $\theta_j^{(k)}$. By setting the learning parameter η, the ANN is trained well and the optimal parameters $\omega_{ij}^{(k)}$ and bias $\theta_j^{(k)}$ are then obtained. Lastly, the option prices are simulated by inputting the asset prices and remainder time into the trained ANN.

Numerical examples are carried for two options, geometric mean payoff functions and arithmetic mean payoff functions. These examples show that the three-layer ANN has powerful capability for multi-asset option pricing. The error and relative errors of ANN options are very small. In these experiments, we use a variable learning rate η, which ensures that our objective functions continue to decline. In addition, we use partial input data to train the network, thereby reducing the amount of computation in ANN.

In future, we will establish a deep learning ANN with a multi-layer (i.e., $p > 3$) structure for option pricing. The challenge is how to compute the partial derivatives with respect to parameters $\omega_{ij}^{(k)}$ and $\theta_j^{(k)}$. Moreover, we will consider the ANN to price multi-asset complex options, such as American options, Asian options and barrier options, which is a bit complicated.

Author Contributions: Methodology, Z.Z.; Software, H.W. and Y.W.; Formal analysis, C.K.; Resources, Y.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by National Natural Science Foundation of China (Grant No. 12171409) and Key Project of Hunan Provincial Department of Education (Grant No. 21A0533).

Data Availability Statement: The authors confirm that the data supporting the findings of this study are available within the article.

Conflicts of Interest: The authors declare that there are no conflicts of interest regarding the publication of this paper.

Appendix A

Appendix A.1. Algorithms

Algorithm A1 Artificial neural network for multi-asset option pricing

```
Step.1  Generate $N$ samples of stock prices and time $\tau$, i.e, $X^{(k)} = \left(x_1^{(k)}, x_2^{(k)}, \cdots, x_d^{(k)}, \tau^{(k)}\right)'$ for $k = 1, 2, \cdots, N$. Assume the first $N' < N$ data with time $\tau^{(k)} = 0, k = 1, 2, \cdots, N'$.
Step.2  Generate randomly initial network weights $\boldsymbol{\omega}^{(i)}$ for $i = 1, 2$ and $\boldsymbol{\theta}^{(j)}$ for $j = 2, 3$. Setting learning rate $\eta$, iteration number $L$ and control error $\varepsilon$.
——      FOR $i = 1 : L$
Step.3      Input the first $N'$ samples into network, then compute error at $\tau = 0$,
            $$ERR^{(payoff)} = \frac{1}{2N'} \sum_{k=1}^{N'} (U^{(3,k)} - y^{(k)})^2.$$
Step.4      Input $N' + 1 \sim N$ samples into network, then compute PDE error,
            $$ERR^{(PDE)} = \frac{1}{2(N - N')} \sum_{k=N'+1}^{N} (ERR_k)^2$$
            with $ERR_k$ being defined by (28).
Step.5      According to (26) and (37), compute adjustment amount $\Delta\boldsymbol{\omega}^{(payoff,j)}$,$\Delta\boldsymbol{\theta}^{(payoff,j)}$,$\Delta\boldsymbol{\omega}^{(PDE,j)}$ and $\Delta\boldsymbol{\theta}^{(PDE,j)}$ .
Step.6      According to (38), update the network's weights $\boldsymbol{\omega}^{(i)}(i = 1, 2)$ and $\boldsymbol{\theta}^{(j)}(j = 2, 3)$ with learning rate $\eta$.
Step.7      If the error $MSE$ (see expressions (19) and (39)) is less than the preset value $\varepsilon$, break.
——      END FOR
Step.8  Output the network $net\left(\boldsymbol{\omega}^{(1)}, \boldsymbol{\omega}^{(2)}, \boldsymbol{\theta}^{(2)}, \boldsymbol{\theta}^{(3)}, \{U^{(1,k)}\}_{k=1}^{N}\right)$ with optimal $\boldsymbol{\omega}^{(i)}$ and $\boldsymbol{\theta}^{(j)}$ and get the trained network.
Step.9  For any input data $\mathbf{X} = [x_1, x_2, \cdots, x_d, \tau]'$, the output network value is
        $$U^3(\mathbf{X}) = net\left(\boldsymbol{\omega}^{(1)}, \boldsymbol{\omega}^{(2)}, \boldsymbol{\theta}^{(2)}, \boldsymbol{\theta}^{(3)}, \mathbf{X}\right).$$
```

Algorithm A2 Monte Carlo algorithm for multi-asset option pricing with arithmetic mean payoff function

```
Step.1 Generate $N$ samples of stock prices $x_{j,i}^{(k)}$ for asset $i = 1, 2, \ldots, d$, price paths $k = 1, 2, \ldots, N$ and discrete time $\tau_j = j\Delta\tau, j = 0, 1, \ldots, M-1$ with $\Delta\tau = T/(M-1)$.
Step.2 Ensure the random numbers $x_{j,i}^{(k)}$ has co-variance $cov = \sqrt{\Delta\tau}\rho_{IJ}\sigma_I\sigma_J$ at each time $\tau_j$ and at each path $k$. We can use the Matlab command 'mvnrnd(zeros(1,d), cov, M)'.
Step.3 Compute the payoff function $P^{(k)} = max\left(0, K - \sum_{i=1}^{d} \alpha_i e^{x_{M-1,i}^{(k)}}\right)$ for each path $k = 1, 2, \ldots, N$ at time $T = \tau_{M-1}$.
Step.4 Compute the simulated option $V(T, \boldsymbol{x}; K, \boldsymbol{\rho}, \boldsymbol{\sigma}) = \frac{1}{Ne^{rT}} \sum_{k=1}^{N} P^{(k)}$.
```

Remark A1. *For geometric mean payoff function, we modify the simulated option* $P^{(k)} = max(0, K - \sum_{i=1}^{d} e^{\alpha_i x_{M-1,i}^{(k)}})$ *at Step 3.*

Appendix A.2. Computational Results

Table A1. Computational results of ANN for geometric mean payoff function with $d = 2$. $[\boldsymbol{x}', \tau]$ is the input data. ANN and Anal. are the ANN solution and analytical solution, respectively. ERR and RE are the absolute error and relative error between ANN solution and analytical solution, respectively. MSE is the error of training.

Input $[\boldsymbol{x}', \boldsymbol{\tau}]$	**ANN**	**Anal.**	**ERR**	**RE**
	$K = 8, MSE = 1.8 \times 10^{-3}$			
$(-3.91, -3.91, +0.40)$	+7.8216	+7.7888	+0.0328	+0.0042
$(-3.91, -1.87, +0.40)$	+7.7861	+7.9078	−0.1217	−0.0156
$(-3.91, +0.17, +0.40)$	+7.6878	+7.7360	−0.0482	−0.0063
$(-3.91, +2.21, +0.40)$	+7.4150	+7.4818	−0.0668	−0.0090
$(-1.87, -3.91, +0.40)$	+7.7861	+7.7835	+0.0027	+0.0003
$(+0.17, +0.17, +0.40)$	+6.6584	+6.6113	+0.0471	+0.0071
$(+2.21, +0.17, +0.40)$	+4.5601	+4.5837	−0.0236	−0.0052
	$K = 10, MSE = 4.3 \times 10^{-4}$			
$(-3.91, -1.87, +0.40)$	+9.7465	+9.7205	+0.0260	+0.0027
$(-3.91, +0.17, +0.40)$	+9.6482	+9.6928	−0.0447	−0.0046
$(-3.91, +2.21, +0.40)$	+9.3754	+9.3301	+0.0452	+0.0048
$(-1.87, -3.91, +0.40)$	+9.7465	+9.7531	−0.0065	−0.0007
$(-1.87, +0.17, +0.40)$	+9.3754	+9.4092	−0.0339	−0.0036
$(-1.87, +2.21, +0.40)$	+8.6188	+8.6425	−0.0237	−0.0028
$(+0.17, -3.91, +0.40)$	+9.6482	+9.6565	−0.0084	−0.0009
$(+0.17, -1.87, +0.40)$	+9.3754	+9.3850	−0.0097	−0.0010
$(+0.17, +0.17, +0.40)$	+8.6188	+8.6722	−0.0535	−0.0062
$(+0.17, +2.21, +0.40)$	+6.5205	+6.5223	−0.0018	−0.0003
$(+2.21, -3.91, +0.40)$	+9.3754	+9.3442	+0.0311	+0.0033
$(+2.21, -1.87, +0.40)$	+8.6188	+8.6436	−0.0248	−0.0029
$(+2.21, +0.17, +0.40)$	+6.5205	+6.5357	−0.0152	−0.0023

Table A2. Computational results of ANN for geometric mean payoff function with $d = 3$.

Input $[x^T, \tau]$	**ANN**	**Anal.**	**ERR**	**RE**
	$K = 8, MSE = 8.1 \times 10^{-4}$			
$(-3.91, -3.91, -3.91, +0.40)$	$+7.8216$	$+7.8601$	-0.0385	-0.0049
$(-3.91, +0.17, +0.17, +0.40)$	$+7.5379$	$+7.4850$	$+0.0529$	$+0.0070$
$(-3.91, +0.17, +2.21, +0.40)$	$+7.2422$	$+7.2064$	$+0.0358$	$+0.0049$
$(-1.87, -3.91, -1.87, +0.40)$	$+7.7637$	$+7.7131$	$+0.0505$	$+0.0065$
$(-1.87, -3.91, +0.17, +0.40)$	$+7.6878$	$+7.6655$	$+0.0222$	$+0.0029$
$(-1.87, -1.87, -3.91, +0.40)$	$+7.7637$	$+7.7690$	-0.0053	-0.0007
$(-1.87, -1.87, -1.87, +0.40)$	$+7.6878$	$+7.7568$	-0.0691	-0.0090
$(-1.87, -1.87, +2.21, +0.40)$	$+7.2422$	$+7.2045$	$+0.0377$	$+0.0052$
$(-1.87, +0.17, -1.87, +0.40)$	$+7.5379$	$+7.5624$	-0.0245	-0.0032
$(-1.87, +0.17, +0.17, +0.40)$	$+7.2422$	$+7.1991$	$+0.0431$	$+0.0060$
$(-1.87, +2.21, -1.87, +0.40)$	$+7.2422$	$+7.2154$	$+0.0268$	$+0.0037$
$(+0.17, -3.91, -1.87, +0.40)$	$+7.6878$	$+7.6450$	$+0.0427$	$+0.0056$
$(+0.17, -1.87, -3.91, +0.40)$	$+7.6878$	$+7.6628$	$+0.0250$	$+0.0032$
$(+0.17, -1.87, +0.17, +0.40)$	$+7.2422$	$+7.2991$	-0.0569	-0.0079
$(+0.17, -1.87, +2.21, +0.40)$	$+6.6584$	$+6.6864$	-0.0280	-0.0042
$(+0.17, +2.21, -1.87, +0.40)$	$+6.6584$	$+6.6696$	-0.0112	-0.0017
$(+2.21, -3.91, -1.87, +0.40)$	$+7.5379$	$+7.5412$	-0.0033	-0.0004
$(+2.21, -3.91, +2.21, +0.40)$	$+6.6584$	$+6.7058$	-0.0474	-0.0071
$(+2.21, -1.87, -3.91, +0.40)$	$+7.5379$	$+7.6035$	-0.0655	-0.0087
$(+2.21, -1.87, +2.21, +0.40)$	$+5.5060$	$+5.4715$	$+0.0345$	$+0.0063$
$(-3.91, -3.91, -1.87, +0.40)$	$+9.7625$	$+9.8370$	-0.0745	-0.0076
$(-3.91, -3.91, +0.17, +0.40)$	$+9.7241$	$+9.7084$	$+0.0157$	$+0.0016$
$(-3.91, -3.91, +2.21, +0.40)$	$+9.6482$	$+9.5665$	$+0.0817$	$+0.0085$
$(-3.91, -1.87, -3.91, +0.40)$	$+9.7625$	$+9.8223$	-0.0598	-0.0061
$(-3.91, -1.87, -1.87, +0.40)$	$+9.7241$	$+9.6967$	$+0.0274$	$+0.0028$
$(-3.91, +0.17, -3.91, +0.40)$	$+9.7241$	$+9.6769$	$+0.0471$	$+0.0048$
$(-3.91, +0.17, +2.21, +0.40)$	$+9.2026$	$+9.1913$	$+0.0113$	$+0.0012$
$(-3.91, +2.21, +0.17, +0.40)$	$+9.2026$	$+9.1926$	$+0.0099$	$+0.0011$
$(-1.87, -3.91, -3.91, +0.40)$	$+9.7625$	$+9.7628$	-0.0003	-0.0000
$(-1.87, -3.91, +0.17, +0.40)$	$+9.6482$	$+9.7135$	-0.0654	-0.0068
$(-1.87, -3.91, +2.21, +0.40)$	$+9.4983$	$+9.5020$	-0.0037	-0.0004
$(-1.87, -1.87, -1.87, +0.40)$	$+9.6482$	$+9.6656$	-0.0175	-0.0018
$(-1.87, -1.87, +0.17, +0.40)$	$+9.4983$	$+9.5573$	-0.0589	-0.0062
$(-1.87, -1.87, +2.21, +0.40)$	$+9.2026$	$+9.2010$	$+0.0016$	$+0.0002$
$(-1.87, +0.17, -1.87, +0.40)$	$+9.4983$	$+9.4822$	$+0.0161$	$+0.0017$
$(-1.87, +0.17, +2.21, +0.40)$	$+8.6188$	$+8.6905$	-0.0717	-0.0083
$(-1.87, +2.21, -1.87, +0.40)$	$+9.2026$	$+9.2577$	-0.0552	-0.0060
$(-1.87, +2.21, +2.21, +0.40)$	$+7.4664$	$+7.5078$	-0.0415	-0.0056
$(+0.17, -3.91, -3.91, +0.40)$	$+9.7241$	$+9.7144$	$+0.0097$	$+0.0010$
$(+0.17, -3.91, -1.87, +0.40)$	$+9.6482$	$+9.6062$	$+0.0419$	$+0.0043$
$(+0.17, -3.91, +0.17, +0.40)$	$+9.4983$	$+9.5576$	-0.0593	-0.0062
$(+0.17, -1.87, -3.91, +0.40)$	$+9.6482$	$+9.5883$	$+0.0599$	$+0.0062$
$(+0.17, -1.87, -1.87, +0.40)$	$+9.4983$	$+9.4604$	$+0.0379$	$+0.0040$
$(+0.17, -1.87, +0.17, +0.40)$	$+9.2026$	$+9.2699$	-0.0674	-0.0073
$(+0.17, -1.87, +2.21, +0.40)$	$+8.6188$	$+8.5845$	$+0.0343$	$+0.0040$
$(+0.17, +0.17, -3.91, +0.40)$	$+9.4983$	$+9.4349$	$+0.0635$	$+0.0067$
$(+0.17, +0.17, -1.87, +0.40)$	$+9.2026$	$+9.2763$	-0.0738	-0.0080
$(+0.17, +2.21, -3.91, +0.40)$	$+9.2026$	$+9.1713$	$+0.0313$	$+0.0034$
$(+2.21, -3.91, -3.91, +0.40)$	$+9.6482$	$+9.7166$	-0.0684	-0.0071
$(+2.21, -3.91, -1.87, +0.40)$	$+9.4983$	$+9.5741$	-0.0758	-0.0080
$(+2.21, -3.91, +2.21, +0.40)$	$+8.6188$	$+8.5569$	$+0.0619$	$+0.0072$
$(+2.21, -1.87, -3.91, +0.40)$	$+9.4983$	$+9.5719$	-0.0735	-0.0077
$(+2.21, +0.17, -3.91, +0.40)$	$+9.2026$	$+9.2670$	-0.0644	-0.0070

Table A3. Computational results of ANN for geometric mean payoff function with $d = 4$. $[\mathbf{x}', \tau]$ are the input data. ANN and Anal. are the ANN solution and analytical solution, respectively. ERR and RE are the absolute error and relative error between ANN solution and analytical solution, respectively. MSE is the error of training.

Input $[\mathbf{x}^T, \boldsymbol{\tau}]$	**ANN**	**Anal.**	**ERR**	**RE**
	$K = 8, MSE = 1.2 \times 10^{-3}$			
$(-3.91, -3.91, +2.21, -1.87, +0.40)$	+7.6878	+7.6748	+0.0129	+0.0017
$(-3.91, -3.91, +2.21, +0.17, +0.40)$	+7.5854	+7.6169	−0.0314	−0.0041
$(-3.91, -3.91, +2.21, +2.21, +0.40)$	+7.4150	+7.4759	−0.0609	−0.0082
$(-3.91, -1.87, -3.91, -3.91, +0.40)$	+7.8083	+7.7742	+0.0341	+0.0044
$(-3.91, -1.87, -3.91, -1.87, +0.40)$	+7.7861	+7.7877	−0.0016	−0.0002
$(-3.91, -1.87, -1.87, -1.87, +0.40)$	+7.7492	+7.7526	−0.0034	−0.0004
$(-3.91, -1.87, -1.87, +0.17, +0.40)$	+7.6878	+7.7240	−0.0363	−0.0047
$(-3.91, -1.87, -1.87, +2.21, +0.40)$	+7.5854	+7.5666	+0.0188	+0.0025
$(-3.91, -1.87, +0.17, -3.91, +0.40)$	+7.7492	+7.7100	+0.0393	+0.0051
$(-3.91, -1.87, +0.17, -1.87, +0.40)$	+7.6878	+7.6942	−0.0064	−0.0008
$(-3.91, -1.87, +0.17, +0.17, +0.40)$	+7.5854	+7.6411	−0.0557	−0.0073
$(-3.91, -1.87, +0.17, +2.21, +0.40)$	+7.4150	+7.4632	−0.0482	−0.0065
$(-3.91, -1.87, +2.21, -3.91, +0.40)$	+7.6878	+7.6420	+0.0458	+0.0060
$(-3.91, -1.87, +2.21, -1.87, +0.40)$	+7.5854	+7.5844	+0.0010	+0.0001
$(-3.91, -1.87, +2.21, +0.17, +0.40)$	+7.4150	+7.4604	−0.0455	−0.0061
$(-1.87, -1.87, -3.91, +0.17, +0.40)$	+7.6878	+7.6981	−0.0104	−0.0014
$(-1.87, -1.87, -3.91, +2.21, +0.40)$	+7.5854	+7.4987	+0.0867	+0.0114
$(-1.87, -1.87, -1.87, -3.91, +0.40)$	+7.7492	+7.7232	+0.0261	+0.0034
$(-1.87, -1.87, -1.87, -1.87, +0.40)$	+7.6878	+7.7151	−0.0273	−0.0036
$(-1.87, -1.87, -1.87, +0.17, +0.40)$	+7.5854	+7.6486	−0.0632	−0.0083
$(-1.87, -1.87, -1.87, +2.21, +0.40)$	+7.4150	+7.4078	+0.0071	+0.0010
$(-1.87, -1.87, +0.17, -3.91, +0.40)$	+7.6878	+7.6679	+0.0199	+0.0026
$(-1.87, -1.87, +0.17, -1.87, +0.40)$	+7.5854	+7.6244	−0.0390	−0.0051
$(-1.87, -1.87, +0.17, +0.17, +0.40)$	+7.4150	+7.5068	−0.0918	−0.0124
$(-1.87, -1.87, +0.17, +2.21, +0.40)$	+7.1311	+7.1738	−0.0427	−0.0060
$(-1.87, -1.87, +2.21, -3.91, +0.40)$	+7.5854	+7.5571	+0.0283	+0.0037
$(-1.87, -1.87, +2.21, -1.87, +0.40)$	+7.4150	+7.4324	−0.0174	−0.0024
$(-1.87, -1.87, +2.21, +0.17, +0.40)$	+7.1311	+7.1799	−0.0488	−0.0068
$(-1.87, -1.87, +2.21, +2.21, +0.40)$	+6.6584	+6.6481	+0.0103	+0.0015
$(-1.87, +0.17, -3.91, -3.91, +0.40)$	+7.7492	+7.7233	+0.0259	+0.0033
$(-1.87, +0.17, -3.91, -1.87, +0.40)$	+7.6878	+7.6863	+0.0015	+0.0002
$(+0.17, +0.17, -3.91, +0.17, +0.40)$	+7.4150	+7.3730	+0.0420	+0.0057
$(+0.17, +0.17, -3.91, +2.21, +0.40)$	+7.1311	+7.1468	−0.0157	−0.0022
$(+0.17, +0.17, -1.87, -3.91, +0.40)$	+7.5854	+7.6143	−0.0289	−0.0038
$(+0.17, +0.17, -1.87, -1.87, +0.40)$	+7.4150	+7.5071	−0.0922	−0.0124
$(+0.17, +0.17, -1.87, +0.17, +0.40)$	+7.1311	+7.1774	−0.0463	−0.0065
$(+0.17, +0.17, -1.87, +2.21, +0.40)$	+6.6584	+6.6715	−0.0132	−0.0020
$(+0.17, +0.17, +0.17, -3.91, +0.40)$	+7.4150	+7.4622	−0.0472	−0.0064
$(+0.17, +0.17, +0.17, -1.87, +0.40)$	+7.1311	+7.2425	−0.1114	−0.0156
$(+0.17, +0.17, +0.17, +0.17, +0.40)$	+6.6584	+6.6995	−0.0412	−0.0062
$(+0.17, +0.17, +2.21, -3.91, +0.40)$	+7.1311	+7.1357	−0.0046	−0.0006
$(+0.17, +2.21, +2.21, -1.87, +0.40)$	+5.8711	+5.7729	+0.0982	+0.0167
$(+2.21, -3.91, +0.17, -1.87, +0.40)$	+7.4150	+7.4679	−0.0530	−0.0071
$(+2.21, -3.91, +0.17, +0.17, +0.40)$	+7.1311	+7.2295	−0.0984	−0.0138
$(+2.21, -3.91, +0.17, +2.21, +0.40)$	+6.6584	+6.6742	−0.0158	−0.0024
$(+2.21, -3.91, +2.21, -3.91, +0.40)$	+7.4150	+7.3577	+0.0572	+0.0077
$(+2.21, -3.91, +2.21, -1.87, +0.40)$	+7.1311	+7.0954	+0.0357	+0.0050
$(+2.21, -3.91, +2.21, +0.17, +0.40)$	+6.6584	+6.6215	+0.0369	+0.0055
$(+2.21, -3.91, +2.21, +2.21, +0.40)$	+5.8711	+5.7745	+0.0966	+0.0164
$(+2.21, -1.87, -3.91, -3.91, +0.40)$	+7.6878	+7.6959	−0.0082	−0.0011

Table A3. *Cont.*

Input $[x^T, \tau]$	**ANN**	**Anal.**	**ERR**	**RE**
	$K = 10, MSE = 2.2 \times 10^{-3}$			
$(-3.91, -1.87, -1.87, +0.17, +0.40)$	$+9.6482$	$+9.6399$	$+0.0082$	$+0.0009$
$(-3.91, -1.87, -1.87, +2.21, +0.40)$	$+9.5458$	$+9.6132$	-0.0674	-0.0071
$(-3.91, -1.87, +0.17, -3.91, +0.40)$	$+9.7096$	$+9.6602$	$+0.0494$	$+0.0051$
$(-3.91, +2.21, +0.17, +2.21, +0.40)$	$+8.6188$	$+8.5567$	$+0.0620$	$+0.0072$
$(-3.91, +2.21, +2.21, -3.91, +0.40)$	$+9.3754$	$+9.3482$	$+0.0271$	$+0.0029$
$(-3.91, +2.21, +2.21, -1.87, +0.40)$	$+9.0915$	$+8.9321$	$+0.1594$	$+0.0175$
$(-3.91, +2.21, +2.21, +0.17, +0.40)$	$+8.6188$	$+8.6074$	$+0.0114$	$+0.0013$
$(-1.87, +0.17, -3.91, -1.87, +0.40)$	$+9.6482$	$+9.6558$	-0.0076	-0.0008
$(-1.87, +0.17, -3.91, +0.17, +0.40)$	$+9.5458$	$+9.5892$	-0.0434	-0.0045
$(-1.87, +0.17, -3.91, +2.21, +0.40)$	$+9.3754$	$+9.4482$	-0.0729	-0.0078
$(-1.87, +0.17, -1.87, -3.91, +0.40)$	$+9.6482$	$+9.6601$	-0.0119	-0.0012
$(-1.87, +0.17, -1.87, -1.87, +0.40)$	$+9.5458$	$+9.6054$	-0.0596	-0.0062
$(-1.87, +0.17, -1.87, +0.17, +0.40)$	$+9.3754$	$+9.4869$	-0.1116	-0.0119
$(-1.87, +0.17, -1.87, +2.21, +0.40)$	$+9.0915$	$+9.2449$	-0.1534	-0.0169
$(+0.17, -1.87, +0.17, -3.91, +0.40)$	$+9.5458$	$+9.6244$	-0.0786	-0.0082
$(+0.17, -1.87, +0.17, +2.21, +0.40)$	$+8.6188$	$+8.5543$	$+0.0645$	$+0.0075$
$(+0.17, +0.17, -3.91, -3.91, +0.40)$	$+9.6482$	$+9.6409$	$+0.0073$	$+0.0008$
$(+0.17, +0.17, -3.91, -1.87, +0.40)$	$+9.5458$	$+9.5239$	$+0.0219$	$+0.0023$
$(+0.17, +0.17, -3.91, +0.17, +0.40)$	$+9.3754$	$+9.3272$	$+0.0481$	$+0.0051$
$(+0.17, +0.17, -3.91, +2.21, +0.40)$	$+9.0915$	$+9.1138$	-0.0224	-0.0025
$(+2.21, -3.91, +2.21, -1.87, +0.40)$	$+9.0915$	$+9.0316$	$+0.0599$	$+0.0066$
$(+2.21, -3.91, +2.21, +0.17, +0.40)$	$+8.6188$	$+8.7206$	-0.1018	-0.0118
$(+2.21, -3.91, +2.21, +2.21, +0.40)$	$+7.8315$	$+7.9283$	-0.0968	-0.0124
$(+2.21, -1.87, -3.91, -3.91, +0.40)$	$+9.6482$	$+9.6509$	-0.0028	-0.0003
$(+2.21, -1.87, -3.91, -1.87, +0.40)$	$+9.5458$	$+9.5469$	-0.0011	-0.0001
$(+2.21, -1.87, -3.91, +0.17, +0.40)$	$+9.3754$	$+9.3579$	$+0.0175$	$+0.0019$
$(+2.21, -1.87, -1.87, -3.91, +0.40)$	$+9.5458$	$+9.5765$	-0.0307	-0.0032
$(+2.21, -1.87, -1.87, -1.87, +0.40)$	$+9.3754$	$+9.4073$	-0.0320	-0.0034
$(+2.21, -1.87, +0.17, -3.91, +0.40)$	$+9.3754$	$+9.4532$	-0.0778	-0.0083
$(+2.21, -1.87, +0.17, -1.87, +0.40)$	$+9.0915$	$+9.0265$	$+0.0649$	$+0.0071$
$(+2.21, -1.87, +0.17, +0.17, +0.40)$	$+8.6188$	$+8.5463$	$+0.0725$	$+0.0084$

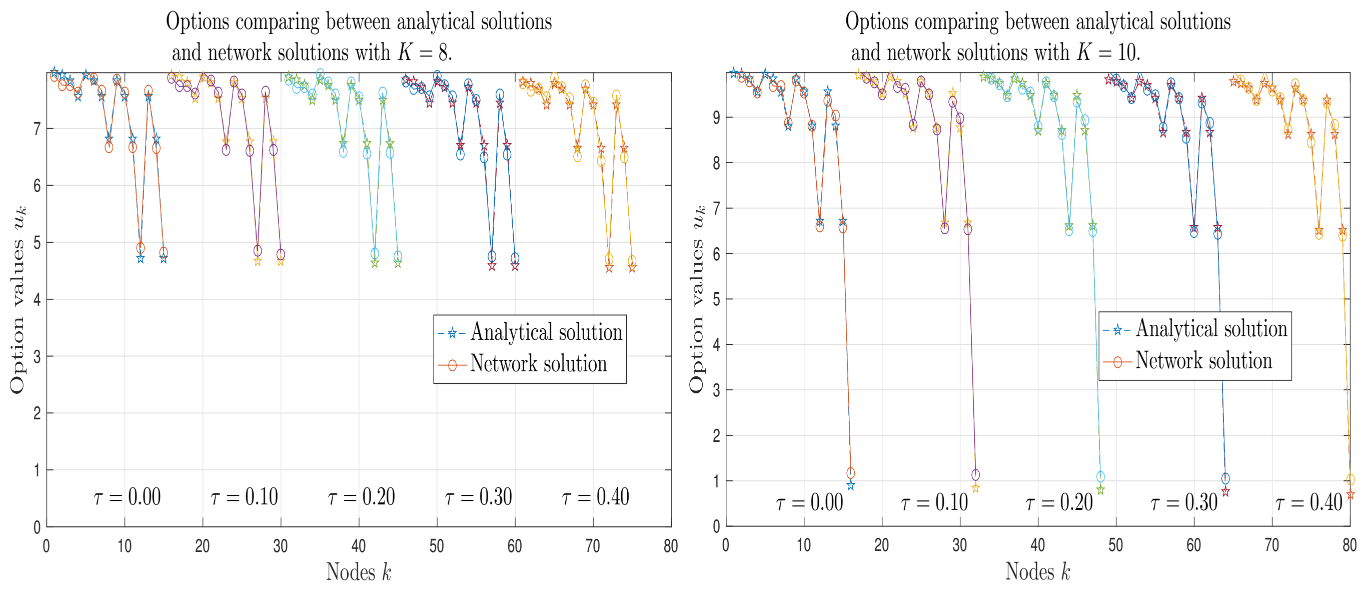

Figure A1. ANN computational options and analytical solutions for geometric mean payoff function. Parameters: $T = 0.4, d = 2$ for $K = 8$ (**left**) and $K = 10$ (**right**). The analytical solution can be seen in (8).

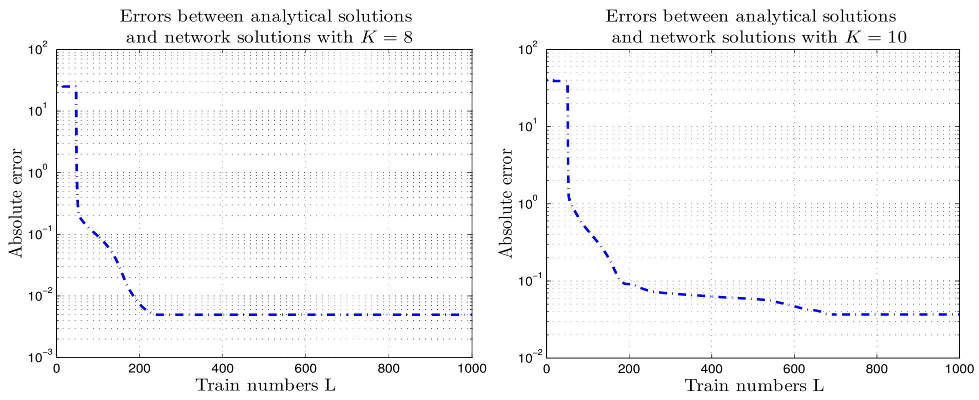

Figure A2. Errors of ANN options vs. training number L with $T = 0.4$ for $K = 8$ (**left**) and $K = 10$ (**right**). The analytical solution can be seen in (8). From this graph, we can see that the error decreases rapidly with the increase in training times.

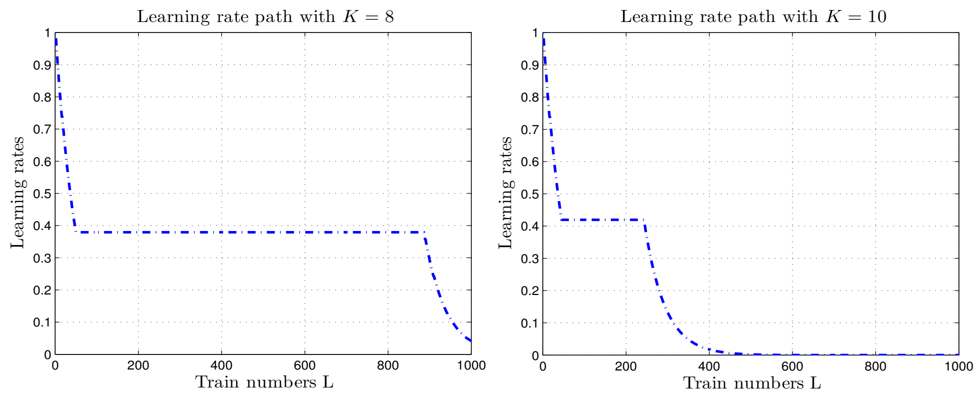

Figure A3. Learning rates vs. training number L with $T = 0.4, d = 2$ for $K = 8$ (**left**) and $K = 10$ (**right**). As can be seen from the figure, the learning rate decreases with the increase in training times.

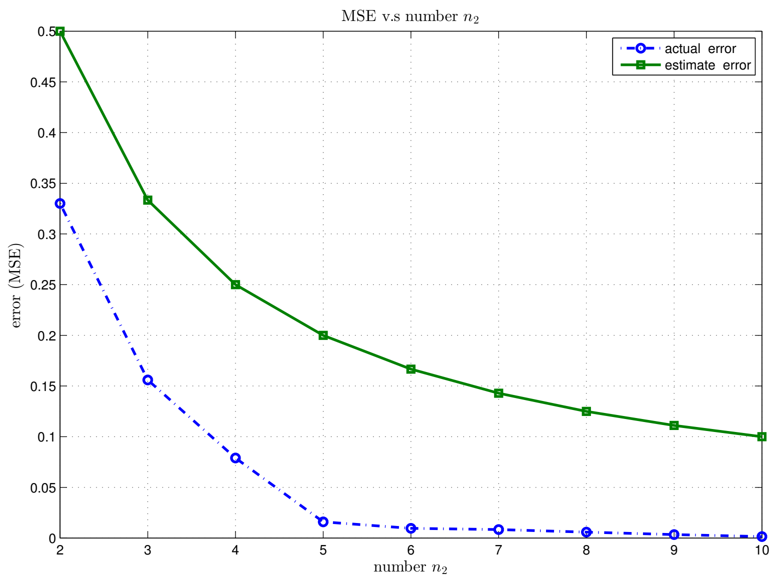

Figure A4. ANN MSE error vs. the number of n_2 with $T = 0.4, d = 2$. The estimate error is set as $ERR(n_2) = 1/n_2$. From this figure, we see the error is about $O\left(\frac{C}{n_2}\right)$ with constant $C > 0$.

Table A4. Computational results of ANN for arithmetic mean payoff function with $d = 2$. $[\boldsymbol{x}', \tau]$ are the input data. ANN and Anal. are the ANN solution and analytical solution, respectively. ERR and RE are the absolute error and relative error between ANN solution and analytical solution, respectively. MSE is the error of training.

Input $[\boldsymbol{x}^T, \tau]$	**ANN**	**Anal.**	**ERR**	**RE**
	$K = 8, MSE = 3.14 \times 10^{-4}$			
$(-3.91, -1.87, +0.40)$	+7.7546	+7.8076	−0.0530	−0.0068
$(-1.87, -3.91, +0.40)$	+7.7546	+7.7609	−0.0063	−0.0008
$(-1.87, +0.17, +0.40)$	+7.1722	+7.1619	+0.0103	+0.0014
$(+0.17, -1.87, +0.40)$	+7.1725	+7.2073	−0.0348	−0.0049
$(+2.21, +0.17, +0.40)$	+2.6955	+2.6926	+0.0029	+0.0011
	$K = 10, MSE = 2.85 \times 10^{-4}$			
$(-3.91, +0.17, +0.40)$	+9.1994	+9.1818	+0.0176	+0.0019
$(+0.17, -3.91, +0.40)$	+9.1997	+9.1792	+0.0205	+0.0022
$(+0.17, -1.87, +0.40)$	+9.1325	+9.1688	−0.0363	−0.0040
$(+0.17, +0.17, +0.40)$	+8.6171	+8.5883	+0.0288	+0.0033
$(+2.21, -3.91, +0.40)$	+5.2365	+5.2662	−0.0297	−0.0057

Table A5. Computational results of ANN for arithmetic mean payoff function with $d = 3$. $[\boldsymbol{x}', \tau]$ are the input data. ANN and Anal. are the ANN solution and analytical solution, respectively. ERR and RE are the absolute error and relative error between ANN solution and analytical solution, respectively. MSE is the error of training.

Input $[\boldsymbol{x}^T, \tau]$	**ANN**	**MC**	**ERR**	**RE**
	$K = 8, MSE = 4.8 \times 10^{-4}$			
$(-3.91, -3.91, -1.87, +0.40)$	+7.7769	+7.8250	−0.0481	−0.0062
$(-3.91, -3.91, +0.17, +0.40)$	+7.4334	+7.4477	−0.0143	−0.0019
$(-3.91, -1.87, -3.91, +0.40)$	+7.7769	+7.7968	−0.0199	−0.0026
$(-3.91, +0.17, -3.91, +0.40)$	+7.4333	+7.3982	+0.0351	+0.0047
$(-3.91, +0.17, +0.17, +0.40)$	+7.0453	+7.0257	+0.0196	+0.0028
$(-3.91, +0.17, +2.21, +0.40)$	+4.4034	+4.4039	−0.0005	−0.0001
$(-1.87, -3.91, -1.87, +0.40)$	+7.7323	+7.7277	+0.0045	+0.0006
$(-1.87, -3.91, +0.17, +0.40)$	+7.3887	+7.3561	+0.0326	+0.0044
$(-1.87, -1.87, -3.91, +0.40)$	+7.7323	+7.7070	+0.0253	+0.0033
$(-1.87, -1.87, +0.17, +0.40)$	+7.3441	+7.3741	−0.0300	−0.0041
$(-1.87, +0.17, -3.91, +0.40)$	+7.3886	+7.3410	+0.0476	+0.0064
$(-1.87, +2.21, +0.17, +0.40)$	+4.3591	+4.3728	−0.0137	−0.0031
$(+0.17, -3.91, -1.87, +0.40)$	+7.3887	+7.3364	+0.0522	+0.0071
$(+0.17, -3.91, +2.21, +0.40)$	+4.4034	+4.3813	+0.0220	+0.0050
$(+0.17, -1.87, -3.91, +0.40)$	+7.3887	+7.3382	+0.0505	+0.0068
$(+0.17, -1.87, -1.87, +0.40)$	+7.3443	+7.3378	+0.0065	+0.0009
	$K = 10, MSE = 1.1 \times 10^{-3}$			
$(-3.91, -1.87, +2.21, +0.40)$	+6.7066	+6.6975	+0.0092	+0.0014
$(-3.91, +0.17, -3.91, +0.40)$	+9.3938	+9.3207	+0.0731	+0.0078
$(-3.91, +2.21, +0.17, +0.40)$	+6.3626	+6.4076	−0.0450	−0.0071
$(-3.91, +2.21, +2.21, +0.40)$	+3.7206	+3.6841	+0.0365	+0.0098
$(-1.87, +0.17, +0.17, +0.40)$	+8.9609	+9.0426	−0.0816	−0.0091
$(+0.17, -3.91, -1.87, +0.40)$	+9.3489	+9.3431	+0.0058	+0.0006
$(+0.17, -3.91, +0.17, +0.40)$	+9.0053	+8.9258	+0.0795	+0.0088
$(+0.17, -3.91, +2.21, +0.40)$	+6.3624	+6.4253	−0.0629	−0.0099
$(+0.17, -1.87, -3.91, +0.40)$	+9.3492	+9.3104	+0.0387	+0.0041
$(+0.17, -1.87, +0.17, +0.40)$	+8.9609	+8.9799	−0.0190	−0.0021

Table A5. *Cont.*

Input $[x^T, \tau]$	**ANN**	**MC**	**ERR**	**RE**
	$K = 10, MSE = 1.1 \times 10^{-3}$			
$(+0.17, +0.17, -1.87, +0.40)$	$+8.9606$	$+8.9319$	$+0.0287$	$+0.0032$
$(+0.17, +0.17, +2.21, +0.40)$	$+5.9758$	$+5.9343$	$+0.0415$	$+0.0069$
$(+0.17, +2.21, -3.91, +0.40)$	$+6.3625$	$+6.4112$	-0.0487	-0.0077
$(+0.17, +2.21, +0.17, +0.40)$	$+5.9747$	$+6.0245$	-0.0498	-0.0083
$(+2.21, -3.91, +0.17, +0.40)$	$+6.3634$	$+6.3904$	-0.0270	-0.0043
$(+2.21, -1.87, +0.17, +0.40)$	$+6.3175$	$+6.3166$	$+0.0009$	$+0.0001$
$(+2.21, +0.17, -3.91, +0.40)$	$+6.3621$	$+6.3359$	$+0.0262$	$+0.0041$
$(+2.21, +0.17, -1.87, +0.40)$	$+6.3182$	$+6.2769$	$+0.0413$	$+0.0065$

Table A6. Computational results of ANN for arithmetic mean payoff function with $d = 4$.

Input $[x^T, \tau]$	**ANN**	**MC**	**ERR**	**RE**
	$K = 8, MSE = 5.84 \times 10^{-4}$			
$(-3.91, -3.91, -3.91, -3.91, +0.40)$	$+7.8216$	$+7.8081$	$+0.0135$	$+0.0017$
$(-3.91, -3.91, -3.91, +0.17, +0.40)$	$+7.5304$	$+7.5918$	-0.0614	-0.0082
$(-3.91, -3.91, -1.87, -1.87, +0.40)$	$+7.7546$	$+7.7980$	-0.0434	-0.0056
$(-3.91, -3.91, -1.87, +0.17, +0.40)$	$+7.4970$	$+7.5068$	-0.0098	-0.0013
$(-3.91, -3.91, +0.17, +2.21, +0.40)$	$+5.2569$	$+5.2893$	-0.0324	-0.0062
$(-3.91, -3.91, +2.21, -3.91, +0.40)$	$+5.5484$	$+5.5281$	$+0.0204$	$+0.0037$
$(-3.91, -3.91, +2.21, -1.87, +0.40)$	$+5.5154$	$+5.5128$	$+0.0026$	$+0.0005$
$(-3.91, -3.91, +2.21, +2.21, +0.40)$	$+3.2779$	$+3.2645$	$+0.0134$	$+0.0041$
$(-3.91, -1.87, -3.91, -3.91, +0.40)$	$+7.7881$	$+7.8140$	-0.0259	-0.0033
$(-1.87, -3.91, -3.91, -3.91, +0.40)$	$+7.7881$	$+7.7979$	-0.0098	-0.0013
$(-1.87, -3.91, -1.87, -3.91, +0.40)$	$+7.7546$	$+7.7171$	$+0.0375$	$+0.0048$
$(-1.87, -3.91, -1.87, +0.17, +0.40)$	$+7.4633$	$+7.5152$	-0.0519	-0.0070
$(-1.87, -3.91, +0.17, -3.91, +0.40)$	$+7.4970$	$+7.4455$	$+0.0515$	$+0.0069$
$(-1.87, -3.91, +0.17, -1.87, +0.40)$	$+7.4634$	$+7.4627$	$+0.0007$	$+0.0001$
$(-1.87, -3.91, +0.17, +0.17, +0.40)$	$+7.2058$	$+7.1455$	$+0.0603$	$+0.0084$
$(-1.87, -3.91, +2.21, +0.17, +0.40)$	$+5.2245$	$+5.2417$	-0.0172	-0.0033
$(-1.87, -1.87, -3.91, -3.91, +0.40)$	$+7.7546$	$+7.8123$	-0.0577	-0.0074
$(-1.87, -1.87, -1.87, -3.91, +0.40)$	$+7.7211$	$+7.7554$	-0.0343	-0.0044
$(-1.87, -1.87, +0.17, -3.91, +0.40)$	$+7.4635$	$+7.5239$	-0.0604	-0.0081
$(-1.87, -1.87, +0.17, +0.17, +0.40)$	$+7.1722$	$+7.2123$	-0.0401	-0.0056
$(+0.17, -1.87, -1.87, +0.17, +0.40)$	$+7.1721$	$+7.2436$	-0.0716	-0.0100
$(+0.17, -1.87, +0.17, +0.17, +0.40)$	$+6.9147$	$+6.9364$	-0.0217	-0.0031
$(+0.17, -1.87, +2.21, +0.17, +0.40)$	$+4.9332$	$+4.9557$	-0.0224	-0.0045
$(+0.17, +0.17, -3.91, -3.91, +0.40)$	$+7.2391$	$+7.2214$	$+0.0176$	$+0.0024$
$(+0.17, +0.17, -3.91, +0.17, +0.40)$	$+6.9485$	$+6.9426$	$+0.0058$	$+0.0008$
$(+0.17, +0.17, -3.91, +2.21, +0.40)$	$+4.9672$	$+4.9248$	$+0.0424$	$+0.0085$
$(+0.17, +0.17, -1.87, -3.91, +0.40)$	$+7.2057$	$+7.2566$	-0.0509	-0.0071
$(+0.17, +0.17, -1.87, +0.17, +0.40)$	$+6.9147$	$+6.9308$	-0.0161	-0.0023
$(+0.17, +0.17, -1.87, +2.21, +0.40)$	$+4.9327$	$+4.9307$	$+0.0019$	$+0.0004$
$(+0.17, +0.17, +0.17, -3.91, +0.40)$	$+6.9483$	$+6.9758$	-0.0276	-0.0040
$(+0.17, +0.17, +0.17, -1.87, +0.40)$	$+6.9144$	$+6.9505$	-0.0361	-0.0052
$(+0.17, +0.17, +0.17, +0.17, +0.40)$	$+6.6572$	$+6.6243$	$+0.0329$	$+0.0049$
$(+0.17, +0.17, +0.17, +2.21, +0.40)$	$+4.6761$	$+4.7129$	-0.0368	-0.0079
$(+0.17, +0.17, +2.21, -3.91, +0.40)$	$+4.9654$	$+4.9871$	-0.0217	-0.0044
$(+0.17, +0.17, +2.21, -1.87, +0.40)$	$+4.9319$	$+4.9594$	-0.0275	-0.0056
$(+2.21, +0.17, -3.91, +0.17, +0.40)$	$+4.9657$	$+4.9267$	$+0.0390$	$+0.0079$
$(+2.21, +0.17, +0.17, -3.91, +0.40)$	$+4.9660$	$+4.9243$	$+0.0418$	$+0.0084$
$(+2.21, +0.17, +0.17, -1.87, +0.40)$	$+4.9321$	$+4.9334$	-0.0012	-0.0003
$(+2.21, +0.17, +0.17, +0.17, +0.40)$	$+4.6752$	$+4.6722$	$+0.0030$	$+0.0006$
$(+2.21, +0.17, +2.21, -1.87, +0.40)$	$+2.9518$	$+2.9571$	-0.0053	-0.0018
$(+2.21, +0.17, +2.21, +0.17, +0.40)$	$+2.6944$	$+2.6819$	$+0.0126$	$+0.0047$
$(+2.21, +2.21, -3.91, +0.17, +0.40)$	$+2.9846$	$+2.9906$	-0.0060	-0.0020

Table A6. *Cont.*

Input $[x^T, \tau]$	**ANN**	**MC**	**ERR**	**RE**
	$K = 10, MSE = 8.3 \times 10^{-4}$			
$(-3.91, +0.17, -1.87, -1.87, +0.40)$	+9.4237	+9.4729	−0.0492	−0.0052
$(-3.91, +0.17, -1.87, +0.17, +0.40)$	+9.1661	+9.0905	+0.0756	+0.0082
$(-3.91, +0.17, +0.17, -1.87, +0.40)$	+9.1663	+9.1235	+0.0428	+0.0047
$(-3.91, +0.17, +0.17, +2.21, +0.40)$	+6.9271	+6.9686	−0.0415	−0.0060
$(-3.91, +0.17, +2.21, +0.17, +0.40)$	+6.9274	+6.9813	−0.0539	−0.0078
$(-3.91, +2.21, -3.91, +0.17, +0.40)$	+7.2174	+7.2644	−0.0470	−0.0065
$(-3.91, +2.21, +0.17, +0.17, +0.40)$	+6.9261	+6.8745	+0.0516	+0.0074
$(-1.87, -3.91, -3.91, -3.91, +0.40)$	+9.7485	+9.6798	+0.0687	+0.0070
$(-1.87, +0.17, -3.91, -1.87, +0.40)$	+9.4238	+9.4567	−0.0329	−0.0035
$(-1.87, +0.17, -3.91, +2.21, +0.40)$	+7.1845	+7.2401	−0.0556	−0.0077
$(-1.87, +0.17, -1.87, -3.91, +0.40)$	+9.4237	+9.4359	−0.0121	−0.0013
$(-1.87, +0.17, -1.87, +0.17, +0.40)$	+9.1326	+9.1322	+0.0004	+0.0000
$(-1.87, +0.17, +0.17, -1.87, +0.40)$	+9.1326	+9.1468	−0.0142	−0.0016
$(-1.87, +0.17, +2.21, -3.91, +0.40)$	+7.1845	+7.1557	+0.0287	+0.0040
$(-1.87, +2.21, +0.17, +0.17, +0.40)$	+6.8937	+6.8708	+0.0229	+0.0033
$(+0.17, -3.91, -3.91, -3.91, +0.40)$	+9.4908	+9.4429	+0.0479	+0.0050
$(+0.17, -3.91, -3.91, -1.87, +0.40)$	+9.4573	+9.5198	−0.0625	−0.0066
$(+0.17, -3.91, -1.87, -3.91, +0.40)$	+9.4574	+9.4519	+0.0055	+0.0006
$(+0.17, -3.91, -1.87, +0.17, +0.40)$	+9.1662	+9.1604	+0.0058	+0.0006
$(+0.17, -3.91, +0.17, -3.91, +0.40)$	+9.1998	+9.1222	+0.0776	+0.0084
$(+0.17, -3.91, +2.21, -3.91, +0.40)$	+7.2180	+7.2115	+0.0065	+0.0009
$(+2.21, -1.87, -3.91, +0.17, +0.40)$	+7.1853	+7.2001	−0.0149	−0.0021
$(+2.21, -1.87, +0.17, -3.91, +0.40)$	+7.1846	+7.1990	−0.0144	−0.0020
$(+2.21, -1.87, +0.17, +0.17, +0.40)$	+6.8925	+6.9404	−0.0479	−0.0069
$(+2.21, -1.87, +2.21, -3.91, +0.40)$	+5.2034	+5.2511	−0.0477	−0.0092
$(+2.21, +0.17, -3.91, +0.17, +0.40)$	+6.9269	+6.9182	+0.0087	+0.0013
$(+2.21, +0.17, +0.17, -3.91, +0.40)$	+6.9265	+6.9487	−0.0222	−0.0032
$(+2.21, +0.17, +0.17, +0.17, +0.40)$	+6.6357	+6.5998	+0.0359	+0.0054
$(+2.21, +2.21, +0.17, +0.17, +0.40)$	+4.6533	+4.6242	+0.0291	+0.0063
$(+2.21, +2.21, +0.17, +2.21, +0.40)$	+2.6712	+2.6801	−0.0089	−0.0033

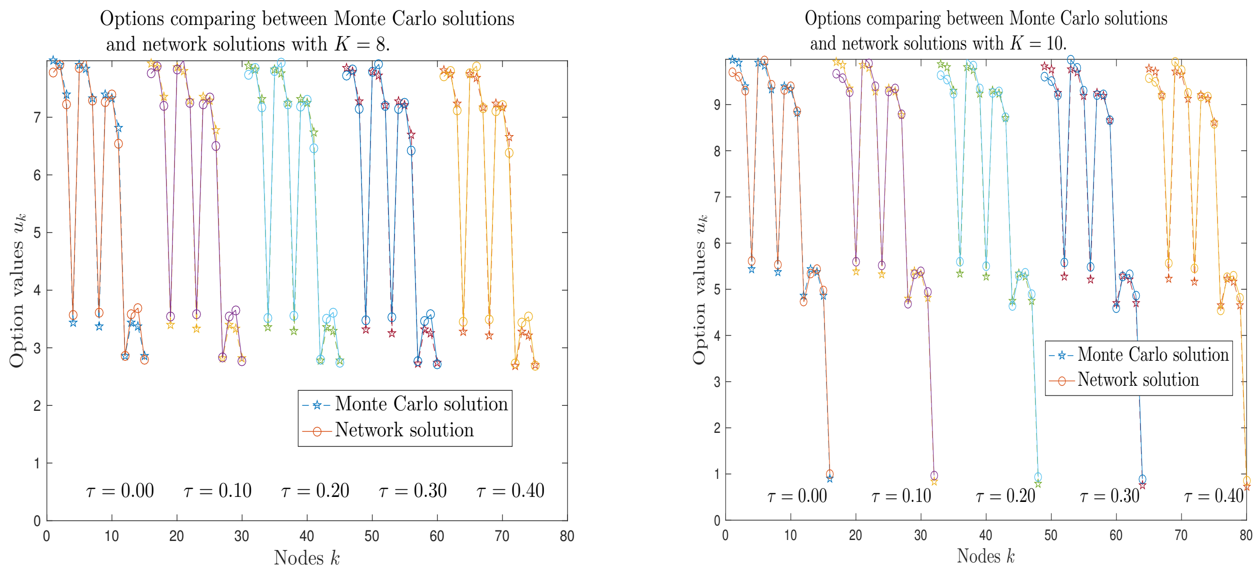

Figure A5. ANN computational options vs. Monte Carlo solutions for arithmetic mean payoff function. Parameters: $T = 0.4, d = 2$ for $K = 8$ (**left**) and $K = 10$ (**right**).

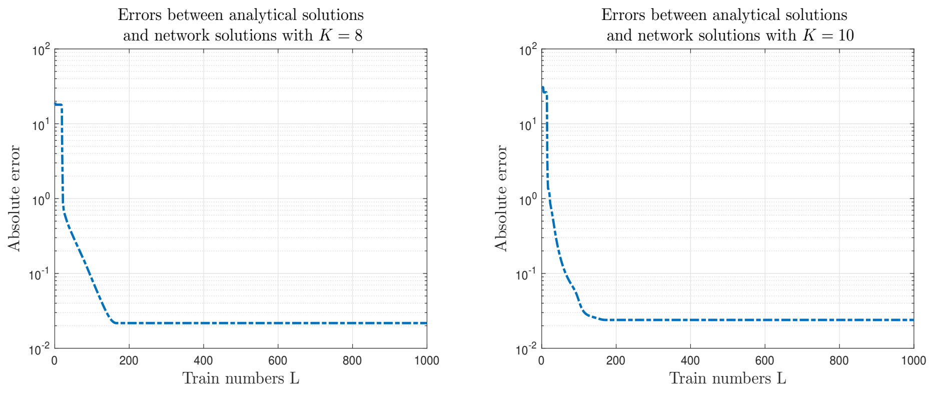

Figure A6. ANN absolute error vs. iteration number L for arithmetic mean payoff function. Parameters: $T = 0.4, d = 2$ for $K = 8$ (**left**) and $K = 10$ (**right**). From this graph, we can see that the error decreases rapidly with the increase in training times.

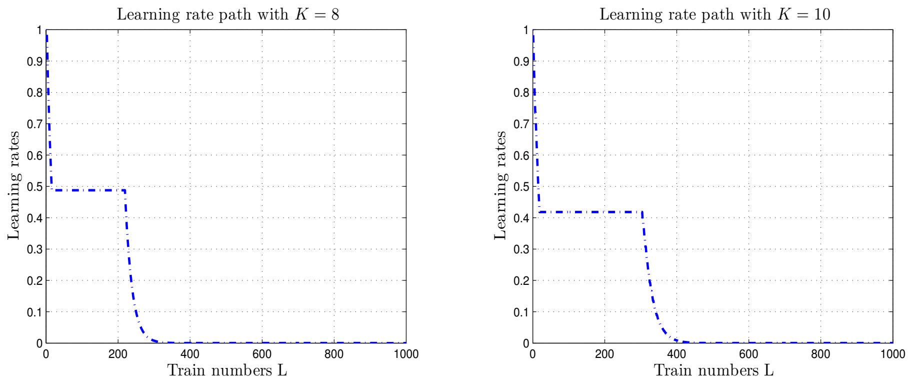

Figure A7. The learning rate vs. iteration number L. Parameters: $T = 0.4, d = 2$ for $K = 8$ (**left**) and $K = 10$ (**right**). As can be seen from the figure, the learning rate decreases with the increase in training times.

References

1. Álvarez, ; D.; González-Rodríguez, P.; Kindelan, M. A local radial basis function method for the Laplace-Beltrami operator. *J. Sci. Comput.* **2021**, *86*, 28. [CrossRef]
2. Banei, S.; Shanazari, K. On the convergence analysis and stability of the RBF-adaptive method for the forward-backward heat problem in 2D. *Appl. Numer. Math.* **2021**, *159*, 297–230. [CrossRef]
3. Bastani, A.F.; Ahmadi, Z.; Damircheli, D. A radial basis collocation method for pricing American options under regime-switching jump-diffusion models. *Appl. Numer. Math.* **2013**, *65*, 79–90. [CrossRef]
4. Bollig, E.; Flyer, N.; Erlebacher, G. Solution to PDEs using radial basis function finite-differences (RBF-FD) on multiple GPUs. *J. Comput. Phys.* **2012**, *231*, 7133–7151. [CrossRef]
5. Fornberg, B.; Lehto, E. Stabilization of RBF-generated finite difference methods for convective PDEs. *J. Comput. Phys.* **2011**, *230*, 2270–2285. [CrossRef]
6. Fornberg, B.; Piret, C. A stable algorithm for at radial basis functions on a sphere. *Siam J. Sci. Comput.* **2007**, *30*, 60–80. [CrossRef]
7. Fornberg, B.; Piret, C. On choosing a radial basis function and a shape parameter when solving a convective PDE on a sphere. *J. Comput. Phys.* **2008**, *227*, 2758–2780. [CrossRef]
8. Larsson, E.; Shcherbakov, V.; Heryudono, A. A least squares radial basis function partition of unity method for solving PDEs. *Siam J. Sci. Comput.* **2017**, *39*, 2538–2563. [CrossRef]
9. Li, H.G.; Mollapourasl, R.; Haghi, M. A local radial basis function method for pricing options under the regime switching model, *J. Sci. Comput.* **2019**, *79*, 517–541. [CrossRef]
10. Shcherbakov, V.; Larsson, E. Radial basis function partition of unity methods for pricing vanilla Basket options. *Comput. Math. Appl.* **2016**, *71*, 185–200. [CrossRef]
11. Zhou, Z.; Wu, H.; Li, Y.; Kang, C.; Wu, Y. Using Time-Space Double Radial Basis Function Method to Solve High-Dimensional PDEs Arising from Multiasset Option Pricing. *Discret. Dyn. Nat. Soc.* **2024**, 5226282.
12. Sunday, E.; Olabisi, U.; Enahoro, O. Analytical solutions of the Black–Scholes pricing model for European option valuation via a projected differential transformation method. *Entropy* **2015**, *17*, 7510–7521. [CrossRef]

13. Zhao W.J. An artificial boundary method for American option pricing under the CEV model. *Siam J. Numer. Anal.* **2008**, *46*, 2183–2209.
14. Chiarella, C.; Kang, B.; Meyer, G.H. The numerical solution of the American option pricing problem-finite difference and transform approaches. *World Sci. Books* **2014**, *127*, 161–168.
15. Hout, K.J.I.; Foulon, S. ADI finite difference schemes for option pricing in the Heston model with correlation. *Int. J. Numer. Anal. Model.* **2008**, *7*, 303–320.
16. Hout, K.J.I.; Weideman, J.A.C. A contour integral method for the Black–Scholes and Heston equations. *Siam J. Sci. Comput.* **2011**, *33*, 763–785. [CrossRef]
17. Pang, H.; Sun, H. Fast numerical contour integral method for fractional diffusion equations. *J. Sci. Comput.* **2016**, *66*, 41–66. [CrossRef]
18. Song, L.; Wang, W. Solution of the fractional Black-Scholes option pricing model by finite difference method. In *Abstract and Applied Analysis*; Hindawi Publishing Corporation: Cairo, Egypt; 2013; pp. 1–16.
19. Gabriel, T.A.; Amburgey, J.D.; Bishop, B.L. *CALOR: A Monte Carlo Program Package for the Design and Analysis of Calorimeter Systems*; Osti Information Bridge Server; Oak Ridge National Lab.: Oak Ridge, TN, USA, 2024;. [CrossRef]
20. Gamba A. An extension of least squares Monte Carlo simulation for multi-options problems. In Proceedings of the Sixth Annual International Real Options Conference, Paphos, Cyprus, 4 July 2002; pp. 1–40.
21. Rodriguez, A.L.; Grzelak, L.A.; Oosterlee, C.W. On an efficient multiple time-step Monte Carlo simulation of the SABR model. *Soc. Sci. Electron. Publ.* **2016**, *17*, 1549–1565. [CrossRef]
22. Ma, J.; Zhou, Z. Convergence analysis of iterative Laplace transform methods for the coupled PDEs from regime-switching option pricing. *J. Sci. Comput.* **2018**, *75*, 1656–1674. [CrossRef]
23. Ma, J.; Zhou, Z. Fast Laplace transform methods for the PDE system of Parisian and Parasian option pricing. *Sci. China Math.* **2022**, *65*, 1229–1246. [CrossRef]
24. Panini, R.; Srivastav, R.P. Option pricing with Mellin transforms. *Math. Comput. Model.* **2004**, *40*, 43–56. [CrossRef]
25. Zhou, Z.; Ma, J.; Sun, H. Fast Laplace transform methods for free-boundary problems of fractional diffusion equations. *J. Sci. Comput.* **2018**, *74*, 49–69. [CrossRef]
26. Wu, H.; Zhou, Z.; Kang, C. Option pricing by willow tree method for generalized Hyperbolic Lev́y Processes. *J. Math.* **2023**, *2003*, 996556.
27. Anderson, D.; Ulrych, U. Accelerated American option pricing with deep neural networks; *Quant. Financ. Econ.* **2022**, *7*, 207–228. 10.2139/ssrn.4000756. [CrossRef]
28. Carverhill, A.P.; Cheuk, T.H.F. Alternative neural network approach for option pricing and hedging. *Soc. Sci. Electron. Publ.* **2024**, 1–17. [CrossRef]
29. Gan, L.; Liu, W. Option pricing based on the residual neural network. *Comput. Econ.* **2023**, *63*, 1327–1347. [CrossRef]
30. Glau, K.; ; Wunderlich, L. Neural network expression rates and applications of the deep parametric PDE method in counterparty credit risk. *Ann. Oper. Res.* **2024**, *336*, 331–357. [CrossRef]
31. He, W.; Guan, M. Parameter estimation method of option pricing model based on convolutional neural network in high frequency financial trading. *Ann. Oper. Res.* **2022** . [CrossRef]
32. Kapllani, L.; Teng, L. Deep learning algorithms for solving high-dimensional nonlinear backward stochastic differential equations. *Discret. Contin. Dyn. Syst. Ser. B* **2024**, *29*, 1695–1729. [CrossRef]
33. Lee, Y.; Son, Y. Predicting arbitrage-free American option prices using artificial neural network with pseudo inputs. *Ind. Eng. Manag. Syst.* **2021**, *20*, 119–129. [CrossRef]
34. Mary, M.; Salchenberger, L. A neural network model for estimating option prices. *Appl. Intell.* **1993**, *3*, 193–206.
35. Shvimer, Y.; Zhu, S.P. Pricing options with a new hybrid neural network model. *Expert Syst. Appl.* **2024**, *251*, 123979. [CrossRef]
36. Teng, Y.; Li, Y.; Wu, X. Option volatility investment strategy: The combination of neural network and classical volatility prediction model. *Discret. Dyn. Nat. Soc.* **2022**, *2022*, 8952996. [CrossRef]
37. Tung, W.L.; Quek, C. Financial volatility trading using a self-organising neural-fuzzy semantic network and option straddle-based approach *Expert Syst. Appl.* **2011**, *38*, 4668–4688.
38. Umeorah, N.; Mashele, P.; Agbaeze, O.M.J.C. Barrier Options and Greeks: Modeling with Neural Networks. *Axioms* **2023**, *12*, 384. [CrossRef]
39. Wang H. Nonlinear neural network forecasting model for stock index option price: Hybrid GJR-CGARCH approach. *Expert Syst. Appl.* **2009**, *36*, 564–570. [CrossRef]
40. Jiang L.S. *Mathematical Models and Method of Option Pricing (Chinese Edition)*; Higher Education Press: Beijing, China, 2008.