Journal of Applied Analysis and Computation
Volume 16, Number 6, December 2026, 2819–2840
Website:http://www.jaac-online.com
DOI:10.11948/20250377

SOLVING STOCHASTIC INVENTORY MODEL EQUATION WITH DEEP NEURAL NETWORKS*

Shijing Si[1], Yijin Gao[1], Haixia Sun[1,†] and Aifan Ling[1]

Abstract The joint optimization of dynamic pricing and inventory control in stochastic inventory systems has been extensively investigated in the previous literature before. A general common approach is to transform the dynamic system into a time-independent ordinary differential equation that represents the stable state. In this study, we extend the model to incorporate the general case with time-dependent dynamics. Consequently, the problem can be reformulated as solving a corresponding partial differential equation (PDE) that lacks an analytical solution. However, assuming an analytical solution encounters certain limitations: (1) Ensuring the accurate existence and uniqueness of the solution is challenging, and (2) completely solving the reduced ordinary differential equation system becomes difficult from theoretical sense. To address these challenges, efficient numerical methods are commonly employed. Nevertheless, for high-dimensional problems, numerical methods like finite element or finite difference methods tend to be computationally slow and suffer from accuracy loss. To overcome these issues, we propose the use of a deep learning technique named Physics-informed Neural Networks (PINN) to solve this complex system, which is the first application of PINN application in this research topic. Simulated examples are presented to validate the usefulness of PINN methods.

Keywords Stochastic inventory model, partial differential equation, deep learning method, physics-informed neural network.

MSC(2010) 90B05, 49N90, 68T07.

1. Introduction

Price is one of the key factors that can be manipulated to have influence on product demand. The quick evolution of the society makes the dynamic pricing much easier. In today's competitive global market, the dynamic pricing is an important topic of manufacturing enterprises. Thus pricing and production strategies are the critical factors to determine profitability. In practice, apart from the existing direct factors of the inventory level, the inventory will also suffer from the external stochastic disturbances, which can be described as the Brownian motion inside the equation. The effect of deterioration of inventory is very important in the inventory systems.

†The corresponding author.
[1]School of Economics and Finance, Shanghai International Studies University, Shanghai 201620, China
*The authors were supported by the 2024 Shanghai Educational Science Research Project (Special Project for Philosophy and Social Sciences Research in Shanghai Higher Education Institutions) entitled "Digital Technology Empowering the Great Founding Spirit of the Communist Party of China" (Grant No. 2024ZSW009), and the Fundamental Research Funds for the Central Universities (Grant No. 41005246).
Email: shijing.si@outlook.com(S. Si), yjgao@shisu.edu.cn(Y. Gao), sunhx@shisu.edu.cn(H. Sun), aiffling@163.com(A. Ling)

Thus in our paper, we discuss their relations from the mathematical view. The basic idea is applying partial differential equation method for the system.

In the paper [20], they consider the joint dynamic pricing and inventory control policy for a stochastic inventory system with perishable products. In their model, the expected value of the discounted profits from time t to infinity is time-independent. Thus they define the value function as $V(x)$ instead of $V(x,t)$. Then the corresponding Hamilton-Jacobi-Bellman (HJB) equation will be modified as an ordinary differential equation (ODE) with analytical solution. Mostly in the previous literature, they tell us the story about the stable state for the inventory system: $\frac{\partial V}{\partial t} = 0$ and we can not see the system evolution during the fixed time period. In our paper, we extend the initial ordinary differential equation to a partial differential equation (PDE) model with time-dependent situation. A PDE with the value function $V(x,t)$ is derived with two variables x and t. Generally, we can assume the solution with some specific forms for the analytical solution: Linear, quadratic, exponential or cubic. To the best knowledge as we know, it has some disadvantages in real practice: (1) We do not know the uniqueness or existence of the analytical solution mathematically. It is inaccurate to give such solution directly. (2) To solve the quadratic function, we need to solve three ODEs with complicated formulas which are expensive computing.

The advantage of guessing solution with the specific form is modifying the PDE problem into corresponding ODE system which has analytical solutions. However, we are not sure about the uniqueness property of the solution. Thus the quadratic form solution is just one possibility for the system and it lacks mathematical logic in some sense. Still, the ODE system has several equations and some of them are not easy to be solved with analytical solution because of the existence of integration. It is necessary to propose some efficient numerical methods to tackle such problem. Finite difference method (FDM) or the finite element method (FEM) generally are the classical strategies to deal with partial derivatives inside the differential equation. For the linear PDE, FDM can be applied efficiently since the linear system can be solved numerically with algorithms. But in our inventory model problem, the PDE contains the nonlinear term $(V_x)^2$ which can not be discretized with linear matrix. In [7], we have presented efficient Hopf-Cole transformation and operator splitting method. But it still has some constraints: Numerical method is more accurate than guessing solution, but generally it needs more computing time with complicated algorithm; We can not extend the algorithm to higher dimensions which have more applications in the realistic models with more unsolvable variables. Thus alternative method is necessary for the problem.

Deep neural networks (DNNs) has achieved great progress in the artificial intelligence areas [19], especially in computer vision [8], natural language processing [14], and many other areas [12, 23]. They also have shown good promise in many scientific research problems that require intensive computing capacity and extensive domain expertise [4, 11, 16]. DNNs utilize multiple layers of interconnected neurons to automatically learn important features from high-dimensional parameter spaces. By performing an optimization process based on the loss function, the network model brings the promise of a powerful approach to approximate the complex and nonlinear mapping relations of the input-output model. Theorems in [22] proved that the universal function approximation capabilities of neural networks open a new way to obtain the latent solutions of PDE systems.

Recently, there has been a growing interest in utilizing DNNs to address the challenges associated with solving PDEs. [26] introduced a novel approach known as physics-informed neural networks (PINNs) to tackle both forward and inverse problems involving PDEs. The key idea

behind PINNs is to incorporate the governing equations, as well as the initial and boundary conditions, into the loss function as penalizing terms [18]. This strategy aims to effectively constrain the space of latent solutions. Subsequently, the variables (weights and biases) within the network are updated through an optimization process, typically employing methods such as gradient descent or quasi-Newton techniques, to minimize the defined loss function. Once appropriately trained, the resulting network can serve as a data-free universal function approximator [9] that inherently encodes the underlying physical laws as prior knowledge, offering solutions to PDE systems.

Following [26] and [27], many research have demonstrated the potential of this methodology across various scenarios [2, 10, 24]. Python modules, such as DeepXDE [21], have also been developed to facilitate the solution of differential equations. In this paper, we explore the application of PINNs to stochastic inventory models, which is the first attempt in this research field.

The rest of this paper is organized as follows. In Section 2, we review the basic stochastic equation raised in the inventory model, a corresponding PDE is derived through the optimal problem from Ito formula and the analytical solution can be found with the assumption that the solution satisfies a quadratic concave form. PINN algorithm is introduced and applied in Section 3. Efficient simulation examples are presented in Section 4 to verify the theoretical result. In Section 5, we give the brief perturbation analysis based on the small volatility value assumption. We skip the details and introduce the basic idea. Final Section 6 concludes.

2. Basic model

In this section, we firstly review the basic model following [20] and briefly introduce some assumptions. We consider a manufacturer producing one product and having an inventory warehouse. The demand rate for the product is a linear function $d(t) = \alpha(t) - \beta(t)p(t)$ where $d(t)$ is the demand rate at time t, $p(t)$ is the price of the product, $\alpha(t)$ is the fixed constant determined by the market size and $\beta(t)$ represents the sensitivity of the demand with respect to the price. Let $x(t)$ denote the level of the stock at time t and $u(t)$ is the production rate at time t. By taking into account the stochastic disturbance for the inventory system, the evolution of the inventory model is described by the following stochastic differential equation with the initial condition

$$\begin{cases} dx(t) = (-\theta I(x(t) \geq 0)x(t) + u(t) - \alpha(t) + \beta(t)p(t))\, dt + \sigma(t)dw(t), \\ x(0) = x_0, \end{cases} \tag{2.1}$$

where x_0 denotes the initial inventory level. θ is the deterioration coefficient which represents an intrinsically deterministic property of the product perishing with time. Stochastic term $w(t)$ is a white noise process. Our objective is to seek the joint dynamic pricing and production rate that maximises the total expected discounted profit over the infinite planning horizon. The initial function is considered as the expectation function,

$$E\left[\int_0^\infty e^{-\rho t}\left(p(t)(\alpha(t) - \beta(t)p(t)) - ru^2(t) - h(x(t))\right) dt\right],$$

which describes the total expected discounted profit by adding the price multiplied by the demand of the product, and subtracting all costs over the time horizon. $h(x(t))$ is the inventory holding cost function for positive $x(t)$ and shortage cost for negative $x(t)$. Sometimes it is defined

as $h(x(t)) = h_1x^2$ for $x(t) \geq 0$ and $h(x(t)) = h_2x^2$ for $x(t) \leq 0$. In the numerical simulation, we take $x(t)$ as positive for easy computing. Let $V(x,t)$ denote the expected value of the discounted profits from time t to infinity. The function $V(x,t)$ satisfies following HJB equation:

$$\begin{aligned} &-\rho V + V_t + \max_{u(t),p(t)} [p(t)(\alpha(t) - \beta(t)p(t)) - ru^2(t) - h(x(t)) + (-\theta I(x(t) \geq 0)x(t) \\ &+ u(t) - \alpha(t) + \beta(t)p(t))V_x + \frac{\sigma^2(t)}{2}V_{xx}] = 0. \end{aligned} \tag{2.2}$$

The following lemma give the optimal solution satisfying the HJB equation.

Lemma 2.1. [20] *If there is no constraint on the control variables $p(t)$ and $u(t)$, then for the $x(t) \geq 0$, the optimal pricing and production rate $p^*, u^*(t)$ are*

$$\begin{aligned} p^*(t) &= \frac{2\theta + \rho - \sqrt{(2\theta+\rho)^2 + 16k(t)h_1}}{8k(t)}x(t) - \frac{\alpha(t)(2\theta + \rho - \sqrt{(2\theta+\rho)^2 + 16k(t)h_1})}{8k(t)(\rho + \sqrt{(2\theta+\rho)^2 + 16k(t)h_1})} + \frac{\alpha(t)}{2\beta(t)}, \\ u^*(t) &= \frac{2\theta + \rho - \sqrt{(2\theta+\rho)^2 + 16k(t)h_1}}{8k(t)r}x(t) - \frac{\alpha(t)(2\theta + \rho - \sqrt{(2\theta+\rho)^2 + 16k(t)h_1})}{8k(t)r(\rho + \sqrt{(2\theta+\rho)^2 + 16k(t)h_1})}, \end{aligned} \tag{2.3}$$

and for the case $x(t) < 0$, the $p^*, u^*(t)$ are given,

$$\begin{aligned} p^*(t) &= \frac{\rho - \sqrt{\rho^2 + 16k(t)h_2}}{8k(t)}x(t) - \frac{\alpha(t)(\rho - \sqrt{\rho^2 + 16k(t)h_2})}{8k(t)(\rho + \sqrt{\rho^2 + 16k(t)h_2})} + \frac{\alpha(t)}{2\beta(t)}, \\ u^*(t) &= \frac{\rho - \sqrt{\rho^2 + 16k(t)h_2}}{8k(t)r}x(t) - \frac{\alpha(t)(\rho - \sqrt{\rho^2 + 16k(t)h_2})}{8k(t)r(\rho + \sqrt{\rho^2 + 16k(t)h_2})}, \end{aligned} \tag{2.4}$$

where $k(t) = \frac{\beta(t)}{4} + \frac{1}{4r}$.

Once we restrict the finite time T, the corresponding HJB equation is as follows,

$$\begin{aligned} &-\rho V + V_t + \max_{u(t),p(t)} [p(t)(\alpha(t) - \beta(t)p(t)) - ru^2(t) - h(x(t)) + (-\theta I(x(t) \geq 0)x(t) \\ &+ u(t) - \alpha(t) + \beta(t)p(t))V_x + \frac{\sigma^2(t)}{2}V_{xx}] = 0. \end{aligned} \tag{2.5}$$

Take the first derivative with respect to $u(t), p(t)$ and set them as zero:

$$\begin{cases} \alpha(t) - 2\beta(t)p(t) + V_x\beta(t) = 0, \\ V_x - 2ru(t) = 0, \end{cases} \tag{2.6}$$

thus we have the optimal value of $u^*(t) = \frac{V_x}{2r}, p^*(t) = \frac{V_x}{2} + \frac{\alpha(t)}{2\beta(t)}$. Substitute it into the HJB equation and then derive the partial differential equation

$$V_t + \left(\frac{\beta(t)}{4} + \frac{1}{4r}\right)V_x^2 - \left(\frac{\alpha(t)}{2} + \theta I(x(t) \geq 0)x(t)\right)V_x + \frac{\sigma^2(t)}{2}V_{xx} - h(x(t)) + \frac{\alpha^2(t)}{4\beta(t)} - \rho V = 0, \tag{2.7}$$

with the initial condition $V(x, t = 0) = 0$. Similarly as discussed before, we consider a quadratic concave function candidate as follows,

$$V(x,t) = Q(t)x^2 + R(t)x + S(t).$$

Without loss of generality, we assume $x(t) \geq 0$ in our model and $\alpha(t) = \alpha, \beta(t) = \beta$ constants, then $h(x(t)) = h_1 x^2$ and $I(x(t) \geq 0) = 1$,

$$\begin{aligned} &Q'(t)x^2 + R'(t)x + S'(t) + \left(\frac{\beta}{4} + \frac{1}{4r}\right)(2xQ(t) + R(t))^2 - \left(\frac{\alpha}{2} + \theta x\right)(2xQ(t) + R(t)) \\ &+ \frac{\sigma^2}{2}2Q(t) - h_1 x^2 + \frac{\alpha^2}{4\beta} = 0. \end{aligned} \tag{2.8}$$

Combine the term with respect to the order of x from 2 to 0:

$$\begin{cases} x^2 : Q'(t) + \left(\frac{\beta}{4} + \frac{1}{4r}\right)4Q^2(t) - 2Q(t)\theta - h_1 = 0, \\ x^1 : R'(t) + \left(\frac{\beta}{4} + \frac{1}{4r}\right)4Q(t)R(t) - \alpha Q(t) - \theta R(t) = 0, \\ x^0 : S'(t) + \left(\frac{\beta}{4} + \frac{1}{4r}\right)R^2(t) - \frac{\alpha}{2}R(t) - \sigma^2 Q(t) + \frac{\alpha^2}{4\beta} = 0, \end{cases} \tag{2.9}$$

with the initial conditions: $Q(0) = 0, R(0) = 0, S(0) = 0$. The solution of the ODE: $Q'(t) + aQ^2(t) + bQ(t) + c = 0$ is,

$$Q(x) = \frac{\sqrt{4ac - b^2}\tan\left(\frac{1}{2}\left(k_1\sqrt{4ac - b^2} - x\sqrt{4ac - b^2}\right)\right)}{2a},$$

where $a = \frac{\beta}{4} + \frac{1}{4r}, b = -2\theta, c = -h_1$. The coefficient k_1 can be determined by the initial value $Q(0) = 0 \Rightarrow$

$$\tan\left(\frac{1}{2}k_1\sqrt{4ac - b^2}\right) = \frac{b}{\sqrt{4ac - b^2}},$$

k_1 is solvable. The solution of the ODE: $R'(t) + f(t)R(f) + g(t) = 0$ is,

$$R(t) = e^{-\int_0^t f(\xi)d\xi}\int_0^t -e^{\int_0^\eta f(\xi)d\xi}g(\eta)d\eta,$$

where $f(t) = \left(\frac{\beta}{4} + \frac{1}{4r}\right)4Q(t) - \theta$ and $g(t) = -\alpha Q(t)$. Similarly we can have the analytical value of $S(t)$ once we get $Q(t)$ and $R(t)$,

$$S(t) = \int_0^t M(k)dk,$$

here $M(t) = \left(\frac{\beta}{4} + \frac{1}{4r}\right)R^2(t) - \frac{\alpha}{2}R(t) - \sigma^2 Q(t) + \frac{\alpha^2}{4\beta}$. The numerical calculation of the integration can be solved by Trapezoidal rule or other higher order accuracy quadrature rules. ODE system is easy to solve analytically. Thus we consider solving the equation with the help of computers. Numerical method is the classical way to see the approximation. There exists much literature related to numerical methods solving such nonlinear PDE models raised in finance and engineering. A second-order tridiagonal method for American options under jump-diffusion models is introduced in [3, 17] solving the option pricing with a direct adaptive sparse grid approach; [6] studies Pricing Bermudan options in Lévy process models; [30] presents a high-order front-tracking finite difference method for pricing American options under jump-diffusion

models; [13] shows the second order accurate IMEX methods for option pricing under Merton and Kou jump-diffusion models; [31] gives two-step IMEX BDF method for parabolic integro-differential equations with non-smooth initial data arising in finance. The advantage of such methods is: For the general finite difference or finite element methods, it is easy to construct the linear system through the discretization form. However, some drawbacks exist: (1) It is expensive computing unless we need very refined meshes for increasing computing accuracy. (2) Nonlinear term $(V_x)^2$ happens in the equation and numerical algorithm is difficult for solving higher dimension case (larger than two dimensions). We apply PINN algorithm instead. Firstly, we review the PINN algorithm in the next section.

3. PINN method

Based on the previous analysis, we know the solution with quadratic assumption is reasonable and efficientfrom theoretical view. However, the analytical solution has a few drawbacks in some sense: (1) The formula is complicated computing with given function or coefficients. (2) When guessing the quadratic concave solution form, we are not sure if it is the unique solution or different solutions have different convergence as $t \to \infty$. Then we focus on the following equation

$$u_t + au_x^2 - (b + \theta x)u_x + \frac{\sigma^2}{2}u_{xx} - h_1 x^2 + c = 0, \tag{3.1}$$

where

$$a = \frac{\beta}{4} + \frac{1}{4r},\ b = \frac{\alpha}{2},\ c = \frac{\alpha^2}{4\beta}.$$

Other types of equation can be modified to such equation form with variable changes.

Remark 3.1. Actually such equation has another name in physics or chemistry: Reaction-diffusion equation which has two processes separately. For this paper, we focus on the equation itself without other applications for the length of the paper.

In this section, we elaborate how to leverage PINN to solve the PDE in Eq. 3.1.

3.1. Model architecture

The schematic of a PINN for solving the Eq. (3.1) is shown in the Figure 1. In the set-up dealing with spatio-temporal partial differential equations in Eq. (3.1), the temporal and the spatial coordinates (t, x) are the inputs of the multi-layered perceptron (MLP) that approximates the solution vector of the PDE system. Automatic differentiation (AD) [1] is utilized to differentiate the outputs $\hat{u}$ with respect to the inputs $(t,\ x)$ and formulate the governing equations. AD can be implemented directly from the deep learning framework because it is used to compute the gradients and update the network parameters, i.e., weights $\mathbf{W}$ and biases $\mathbf{b}$, during the training. Subsequently we evaluate the partial derivatives u_t with respect to time t in the period $[0, T]$, u_x and u_{xx} with respect to spatial parameter x in the domain Ω. Let us consider the residual of the partial differential equations $e(t, x)$ as the left-hand side of the Eq. (3.2):

$$e := u_t + au_x^2 - (b + \theta x)u_x + \frac{\sigma^2}{2}u_{xx} - h_1 x^2 + c. \tag{3.2}$$

The MLP $\hat{u}$ is a composition of simple differentiable operations which maps the coordinates to the solution. Therefore, it is possible to obtain derivatives of the outputs with respect to

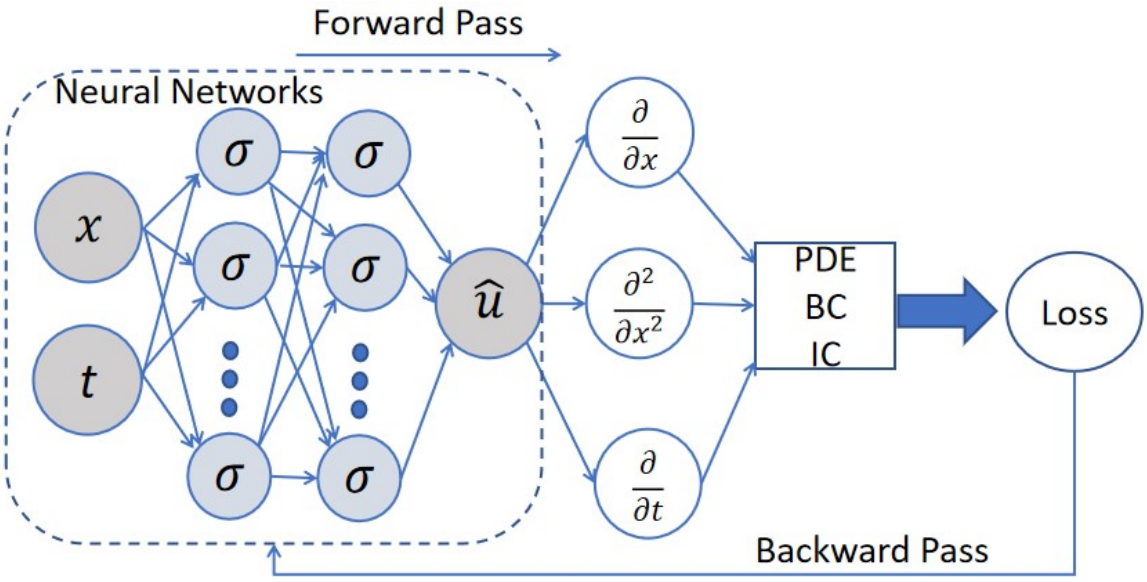

Figure 1. Schematic of a PINN for solving the PDE in Eq. (3.1) with boundary (BC) and initial conditions (IC).

the inputs with the chain rule on $\hat{u}$. Eq. (3.2) can be evaluated by applying the chain rule for differentiating compositions of functions in the MLP using AD, thus yielding the residual of the governing equations. The aforementioned process for computation of the residuals through the implementation of AD is known as the residual network. We use the open-source deep learning framework, PyTorch [25], to implement our PINN models.

3.2. Loss function

The training process of a PINN consists of two parts: Supervised and unsupervised. We refer to supervised learning for those samples for which the target solution is available and unsupervised learning for the samples for which the solution is unknown, while the residual of the governing equations is considered as the loss. Let us consider N_s as the number of points for which we have the solution, e.g., initial and boundary conditions (ICs and BCs) or sparse measurements, and N_e as the number of collocation points for which we calculate the unsupervised loss for training. The unsupervised loss for PDE can be defined as:

$$L_e = \frac{1}{N_e} \sum_{i=1}^{N_e} \left| e(t_e^i, \mathbf{x}_e^i) \right|^2, \tag{3.3}$$

while the supervised loss for ICs and BCs is:

$$L_s = \frac{1}{N_s} \sum_{i=1}^{N_s} \left| \mathbf{u}_s^i - \mathbf{u}(t_s^i, \mathbf{x}_s^i) \right|^2, \tag{3.4}$$

where the subscript s denotes known values. Furthermore, we define the total loss as:

$$L = \alpha L_s + \beta L_e, \tag{3.5}$$

where α and β are the weighting coefficients. The trainable parameters of the neural networks can be learned in a physics-informed manner by minimizing the total loss L. Table 1 presents the procedure for PINN algorithm.

Table 1. The PINN algorithm for solving differential equations.

Step 1. Construct a neural network.
Step 2. Specify the two training sets for the equation and boundary/initial conditions.
Step 3. Specify a loss function by summing the weighted L^2 norm of both the PDE equation and boundary condition residuals.
Step 4. Train the neural network to find the best parameters by minimizing the loss function.

3.3. Training stage

We employ a full-batch training procedure to learn the trainable parameters in the MLP $\hat{u}$. We initiate the training using the Adam optimizer [15] with sufficient number of epochs with the learning rate of 1×10^{-3}. During training, automatic differentiation is used to compute the partial derivatives required for the PDE residual and boundary operator, ensuring the consistency between the DNN approximation and the physical constraints. In the subsequent experiments, the MLP architecture comprises two to four hidden layers, with each layer consisting of 10 to 50 neurons. The network depth and width were manually calibrated during the training phase to minimize the loss function.

4. Simulation examples

In this section, we have three separate examples about the applications of PINN algorithm for classical stochastic inventory equation with the simplified form:

$$u_t + au_x^2 - (b+\theta x)u_x + \frac{\sigma^2}{2}u_{xx} - h_1x^2 + c = 0,$$

where $a, b, \theta, \sigma, h_1, c$ are constants to be determined. In such equation, the nonlinear term is $\mathcal{N}[u;\lambda] = au_x^2 - (b+\theta x)u_x + \frac{\sigma^2}{2}u_{xx} - h_1x^2 + c$. We record down the values of $u(x,t)$ with respect to t and training losses along epochs.

4.1. Example 1

In our first example, we set the coefficient values as: $a = 1, b = h_1 = c = 0, \theta = 1, \sigma^2 = 2$ with zero boundary conditions and initial values: $u(x,0) = x$ or $u(x,0) = x^2$. The figures are including three parts: (1) Loss function Figures 2, 5. (2) For fixed x, the curve of the function $u(x,t)$ Figures 3, 6. (3) For fixed time t, the moving of the $u(x,t)$ Figures 4, 7. The training loss along epochs under various settings are shown in the Table 2. Table 5 shows the $u(x,t)$ under the PDE with b = c = 0 and initial condition $u(x,0) = x$. Theoretically, we know that the value function $u(x,t)$ is decreasing with respect to time t for fixed x, But it has some oscillating results since the second order term. But the tendency is down in some sense. From the figures, the curves fit well.

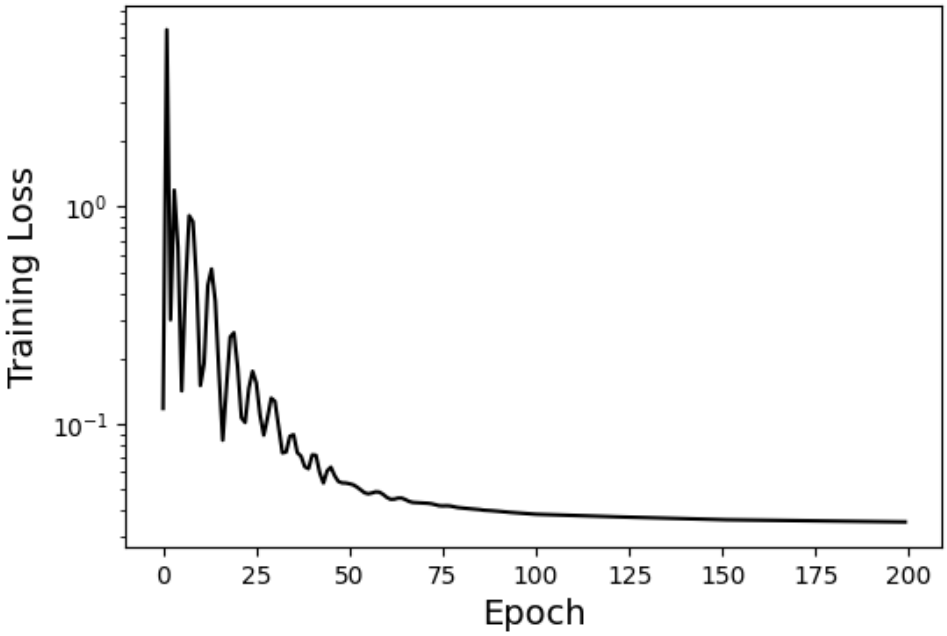

Figure 2. Initial value is x, loss function.

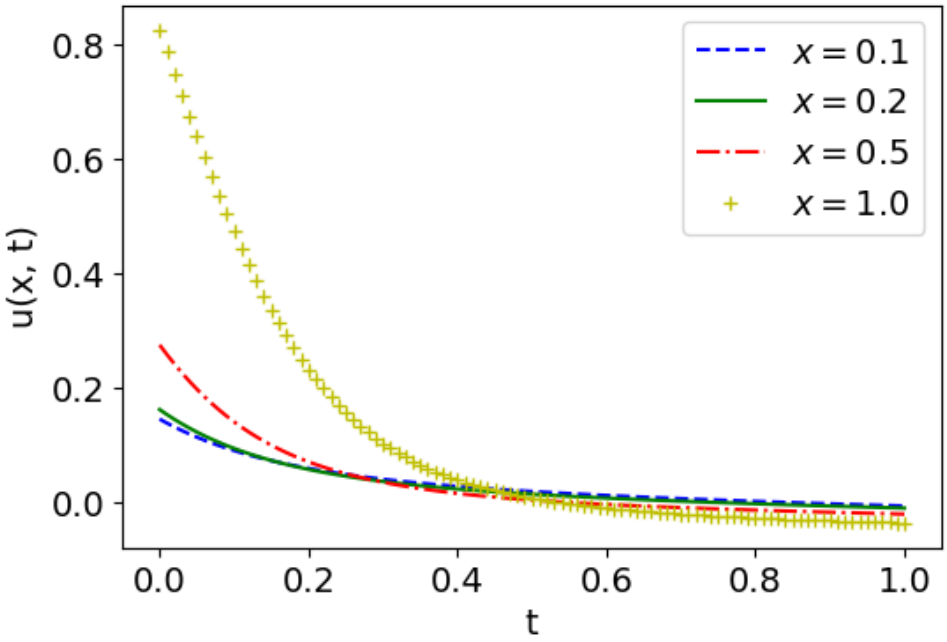

Figure 3. Initial value is x, the $u(x,t)$ with respect to t.

4.2. Effect of network architecture

We have performed a comprehensive ablation study to examine the impact of network depth (number of hidden layers) and width (number of neurons per layer) on the PINN performance. The experiments are conducted under the same settings as Example 1 in Section 4.1, with $a = 1$,

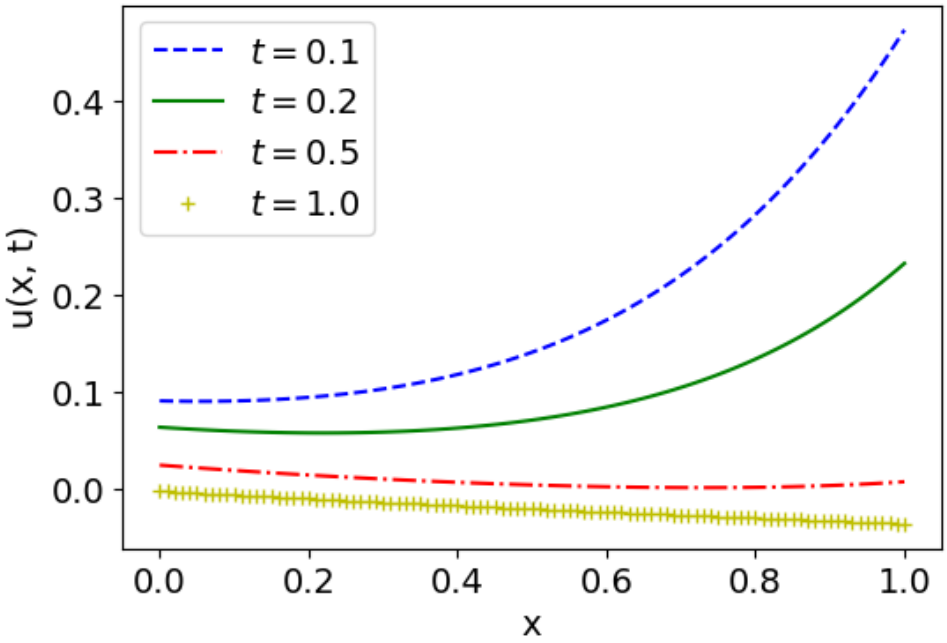

Figure 4. Initial value is x, the $u(x,t)$ with respect to x.

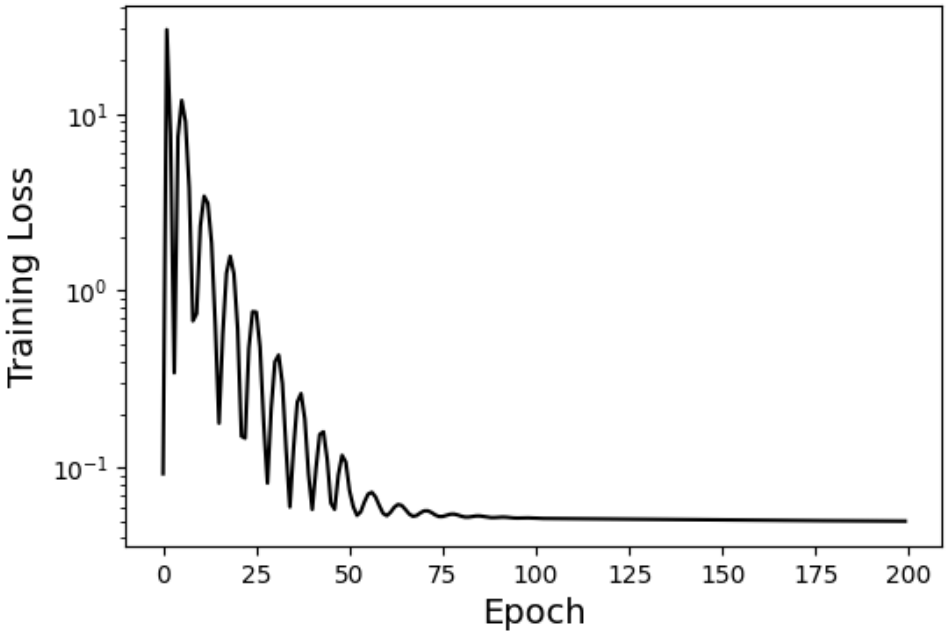

Figure 5. When the initial value is x^2, the loss function versus training epochs.

$b = h_1 = c = 0$, $\theta = 1$, $\sigma^2 = 2$, and initial condition $u(x,0) = x$.

Table 3 and Table 4 display the effects of network depth (Fixed Width = 20 neurons per layer) and network width (Fixed Depth = 4 hidden layers). Key observations are presented as follows.

Table 2. The training loss along epochs under various settings.

Scenarios	5	10	20	50	100	150	200
$b = c = 0,\ u(x,0) = x$	6.53e-1	4.48e-1	2.63e-1	5.33e-2	3.84e-2	3.62e-2	3.53e-2
$b = c = 0,\ u(x,0) = x^2$	7.38e-0	7.43e-1	1.25e-0	1.07e-1	5.19e-2	5.05e-2	4.99e-2
$b = c = 1,\ u(x,0) = x$	2.61e-0	6.30e-1	2.82e-1	1.00e-1	7.13e-2	6.52e-2	6.27e-2
$b = c = 1,\ u(x,0) = x^2$	9.39e-1	9.46e-1	4.35e-1	5.99e-2	4.29e-2	4.21e-2	4.19e-2
$b = c = 0,\ u(x,0) = x, x_{\max} = 2$	2.42e-1	2.84e-1	2.33e-1	2.21e-1	2.20e-1	2.08e-1	1.92e-1

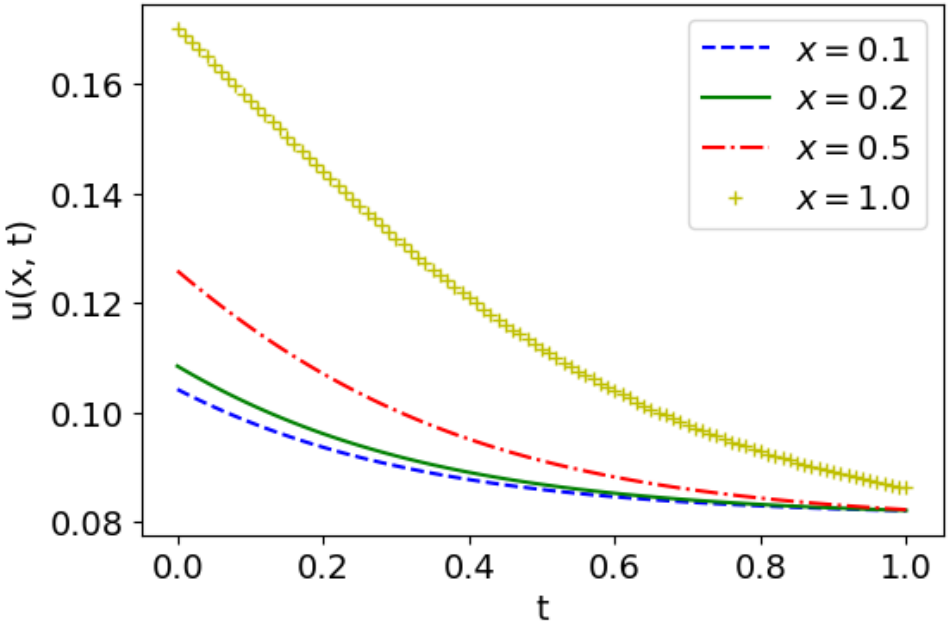

Figure 6. Initial value is x^2, the $u(x,t)$ with respect to t.

(i) **Optimal depth:** The results in Table 3 indicate that increasing network depth from 2 to 4 layers improves the approximation accuracy, with the lowest relative error achieved at 4 hidden layers. However, further increasing depth beyond 4 layers leads to slightly degraded performance and longer training times, likely due to optimization difficulties in deeper networks.

(ii) **Width-performance trade-off:** Table 4 shows that wider networks generally achieve better accuracy, but with diminishing returns and increased computational cost. The improvement from width 20 to width 60 is marginal while training time more than doubles.

(iii) **Practical recommendation:** For the stochastic inventory model considered in this paper, a network architecture with 4 hidden layers and 20-40 neurons per layer provides a good balance between accuracy and computational efficiency. This configuration is used throughout our numerical examples.

The ablation study reveals that the relationship between network capacity and PINN performance is non-monotonic. While increased network capacity provides more expressive power, it also introduces optimization challenges. For the nonlinear PDE arising from stochastic inven-

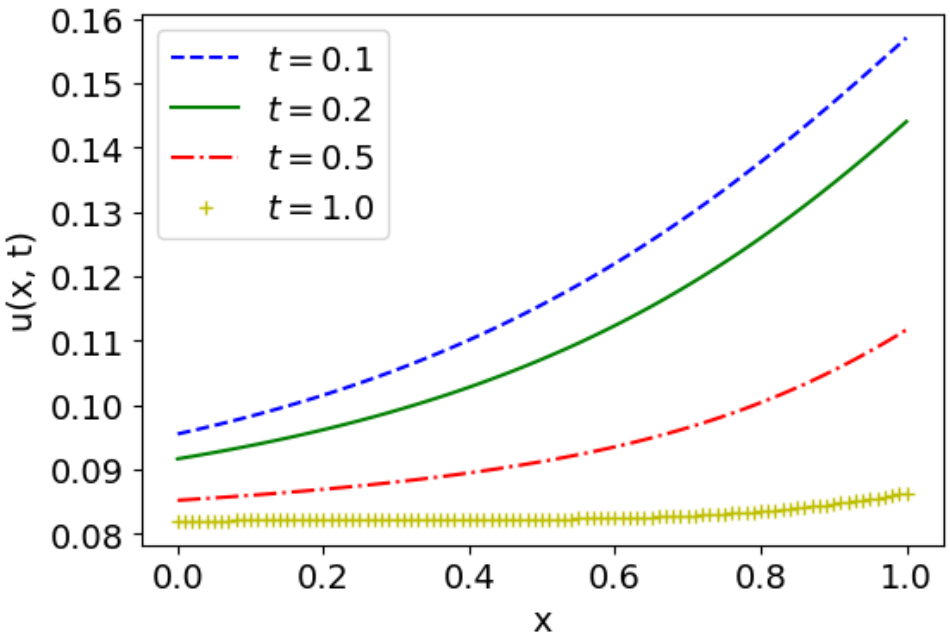

Figure 7. Initial value is x^2, the $u(x,t)$ with respect to x.

Table 3. Performance comparison with different network depths (Fixed Width = 20 neurons per layer).

Depth (Layers)	Final Loss	Training Time (s)
2	5.12×10^{-2}	45.3
3	3.84×10^{-2}	52.7
4	3.53×10^{-2}	61.2
5	3.61×10^{-2}	73.8
6	3.72×10^{-2}	89.4
8	4.15×10^{-2}	125.6

Table 4. Performance comparison with different network widths (Fixed Depth = 4 hidden layers).

Width (Neurons)	Final Loss	Training Time (s)
10	4.87×10^{-2}	38.5
20	3.53×10^{-2}	61.2
40	3.21×10^{-2}	98.7
60	3.08×10^{-2}	142.3
80	3.12×10^{-2}	195.6

tory models, a moderate network depth (4 layers) with sufficient width (20-40 neurons) achieves optimal performance. This finding aligns with the general principle in deep learning that the network architecture should be tailored to the complexity of the target function rather than simply maximizing depth or width.

4.3. Example 2

Similarly, in our second example, we set the coefficient values as nonzero: $a = 1, b = h_1 = c = 1, \theta = 1, \sigma^2 = 2$ with zero boundary conditions and initial values: $u(x, 0) = x$ or $u(x, 0) = x^2$. The figures are including three parts: (1) Loss function Figures 8, 11. (2) For fixed x, the curve of the function $u(x, t)$ Figures 9, 12. (3) For fixed time t, the moving of the $u(x, t)$ Figures 10, 13. Similar as the first example, the curve fits well. Table 8 gives some exact $u(x, t)$ values under the PDE with assumptions $b = c = 0$ and initial condition $u(x, 0) = x^2$. $u(x, t)$ under the PDE with $b = c = 1$ and initial condition $u(x, 0) = x$ is explained in the Table 7.

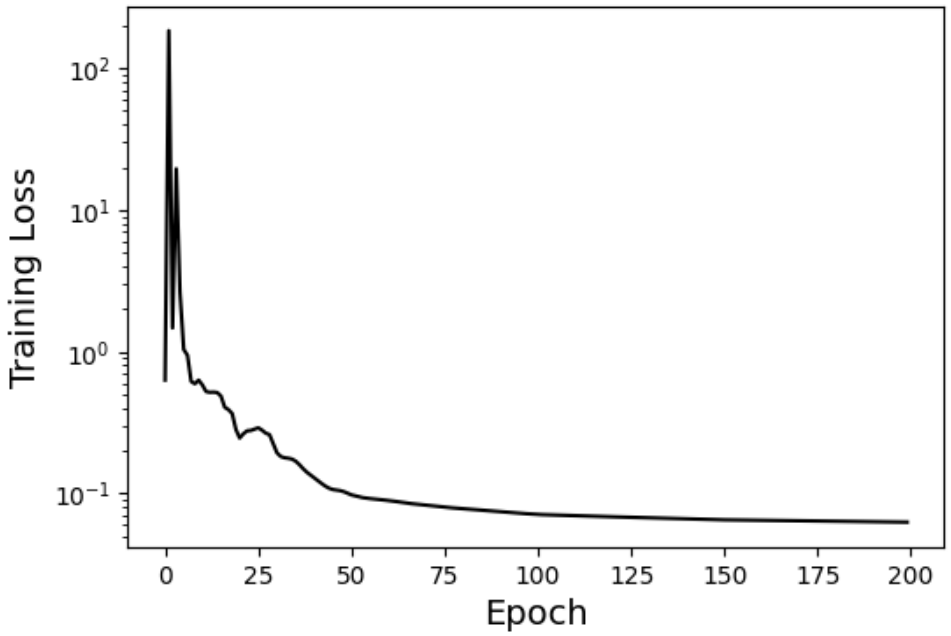

Figure 8. Initial value is x, loss function.

Table 5. $V(x, t)$ under the PDE with $b = c = 0$ and initial condition $V(x, 0) = x$.

$t \setminus x$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.100	0.091	0.091	0.095	0.103	0.118	0.141	0.174	0.220	0.283	0.366	0.473
0.200	0.064	0.060	0.058	0.059	0.063	0.071	0.085	0.105	0.134	0.175	0.233
0.500	0.025	0.019	0.015	0.010	0.007	0.004	0.002	0.002	0.002	0.004	0.008
1.000	-0.002	-0.006	-0.010	-0.013	-0.017	-0.020	-0.023	-0.027	-0.030	-0.033	-0.036

However, there still exists some constraints for PINN method in some sense. Next example shows that: For larger space domain, the training results are not so excellent.

4.4. Example 3

In this example, we extend the domain of x to $x_{\max} = 2$ instead of 1. Other coefficients are the same as the second example. Firstly, from the loss function Figure 14, we will find the curve has more oscillating phenomenon than previous examples. Also the loss value is larger than before

Table 6. $V(x,t)$ under the PDE with $b=c=0$ and initial condition $V(x,0)=x^2$.

$t \setminus x$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.100	0.096	0.098	0.102	0.105	0.110	0.116	0.122	0.129	0.138	0.147	0.157
0.200	0.092	0.094	0.096	0.099	0.103	0.107	0.112	0.119	0.126	0.134	0.144
0.500	0.085	0.086	0.087	0.088	0.089	0.091	0.094	0.096	0.100	0.105	0.112
1.000	0.082	0.082	0.082	0.082	0.082	0.082	0.082	0.083	0.084	0.085	0.086

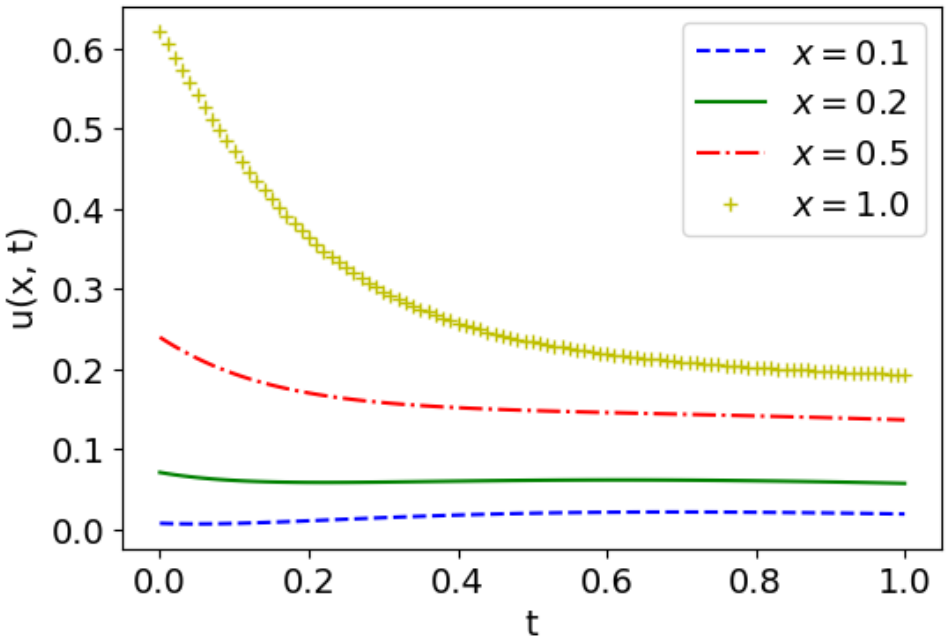

Figure 9. Initial value is x, the $u(x,t)$ with respect to t.

Table 7. $V(x,t)$ under the PDE with $b=c=1$ and initial condition $V(x,0)=x$.

$t \setminus x$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.100	-0.053	0.007	0.061	0.109	0.153	0.194	0.235	0.279	0.330	0.393	0.471
0.200	-0.045	0.011	0.058	0.100	0.137	0.170	0.201	0.232	0.267	0.310	0.364
0.500	-0.028	0.020	0.061	0.096	0.125	0.148	0.168	0.184	0.199	0.215	0.234
1.000	-0.025	0.019	0.057	0.089	0.116	0.137	0.153	0.166	0.176	0.185	0.193

Table 8. $V(x,t)$ under the PDE with $b=c=1$ and initial condition $V(x,0)=x^2$.

$t \setminus x$	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.100	0.082	0.125	0.158	0.184	0.202	0.214	0.222	0.224	0.224	0.220	0.214
0.200	0.094	0.131	0.159	0.180	0.194	0.203	0.207	0.207	0.204	0.199	0.191
0.500	0.059	0.086	0.106	0.121	0.130	0.135	0.135	0.133	0.127	0.120	0.111
1.000	-0.094	-0.073	-0.055	-0.041	-0.031	-0.023	-0.019	-0.016	-0.014	-0.013	-0.012

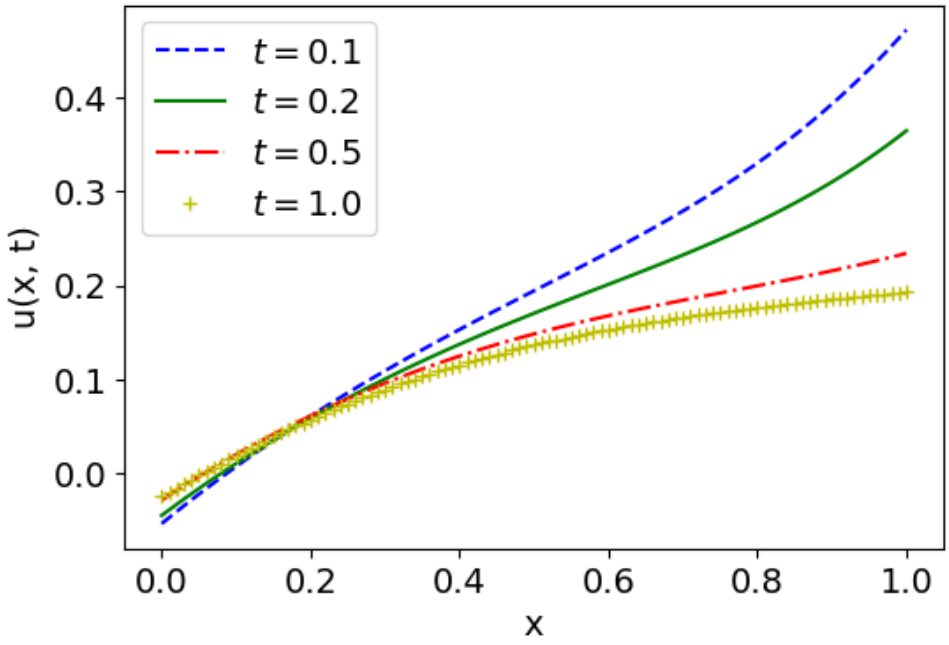

Figure 10. Initial value is x, the $u(x,t)$ with respect to x.

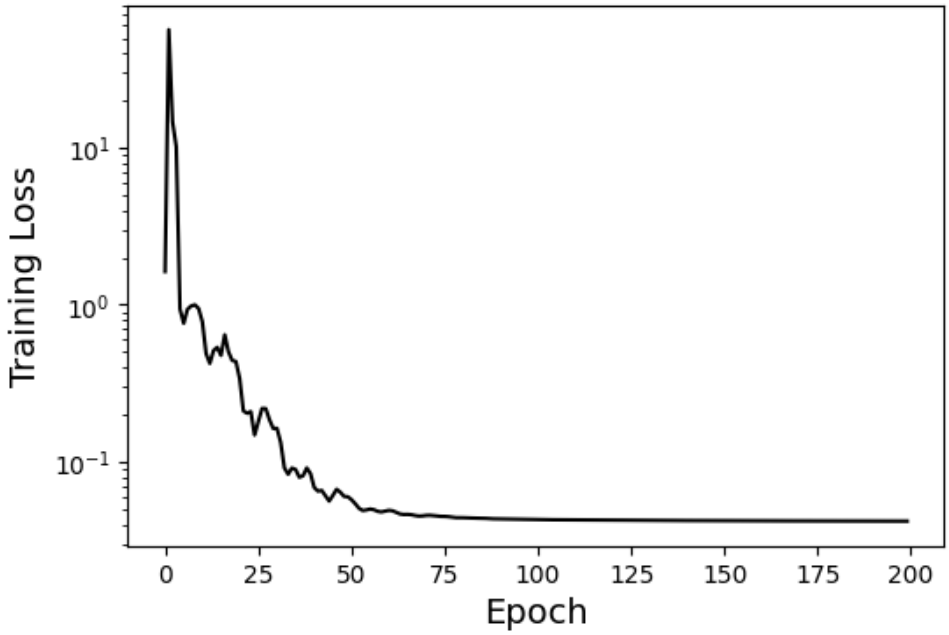

Figure 11. Initial value is x^2, loss function.

which is reasonable as we extend the training domain. The $ux(x,t)$ with respect to x,t are presented in Figures 15, 16. To tackle such problem, we are still searching or advancing more efficient machine learning algorithms.

Remark 4.1. The previous analysis shown in three examples proves the accuracy and efficiency of the PINN algorithm.

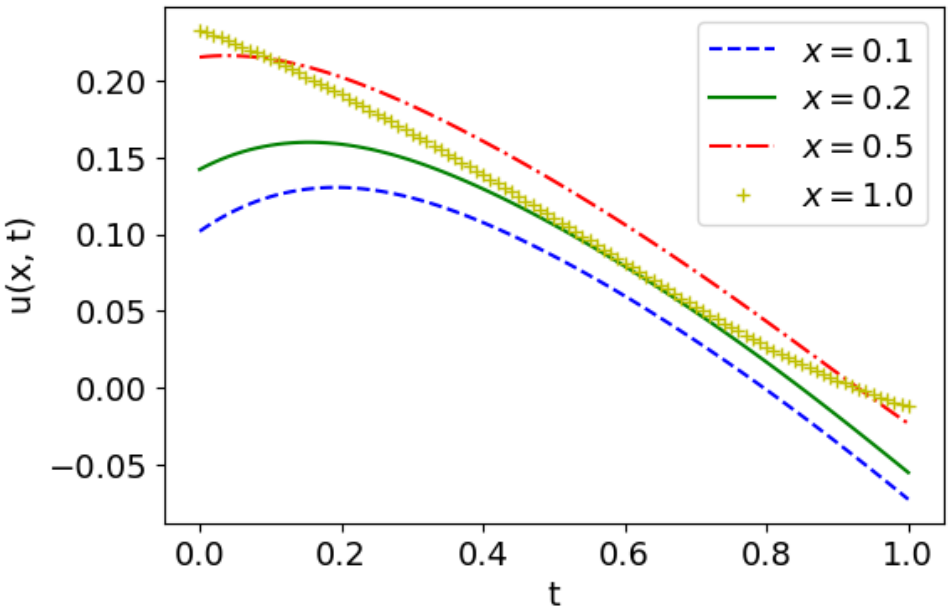

Figure 12. Initial value is x^2,the $u(x,t)$ with respect to t.

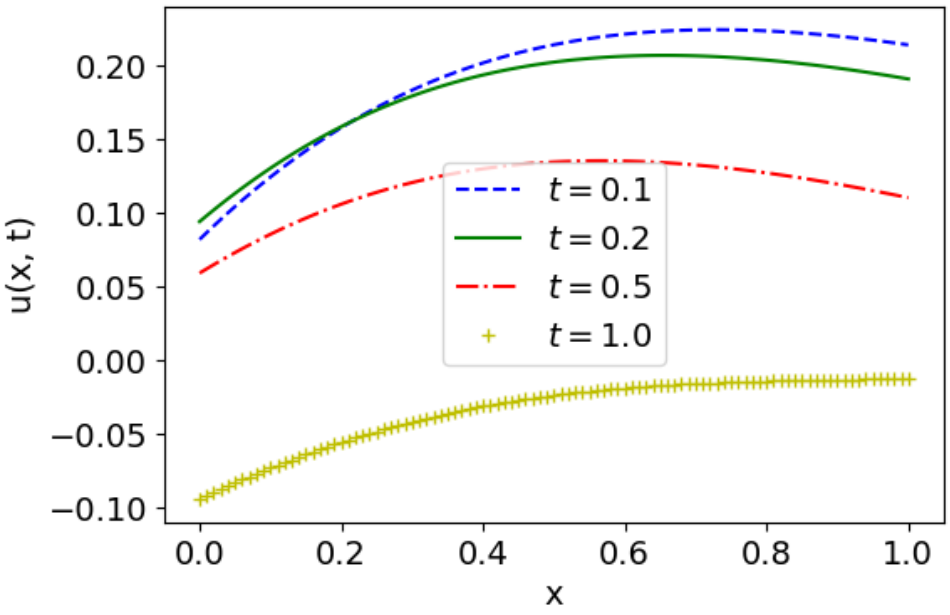

Figure 13. Initial value is x^2, the $u(x,t)$ with respect to x.

5. Perturbation analysis

As we know in the real economy model, the policy change will have impact on the value of coefficients. How does the system change as we modify the equation? In this section, we consider the perturbation theory in the model and derive its corresponding "first order" error

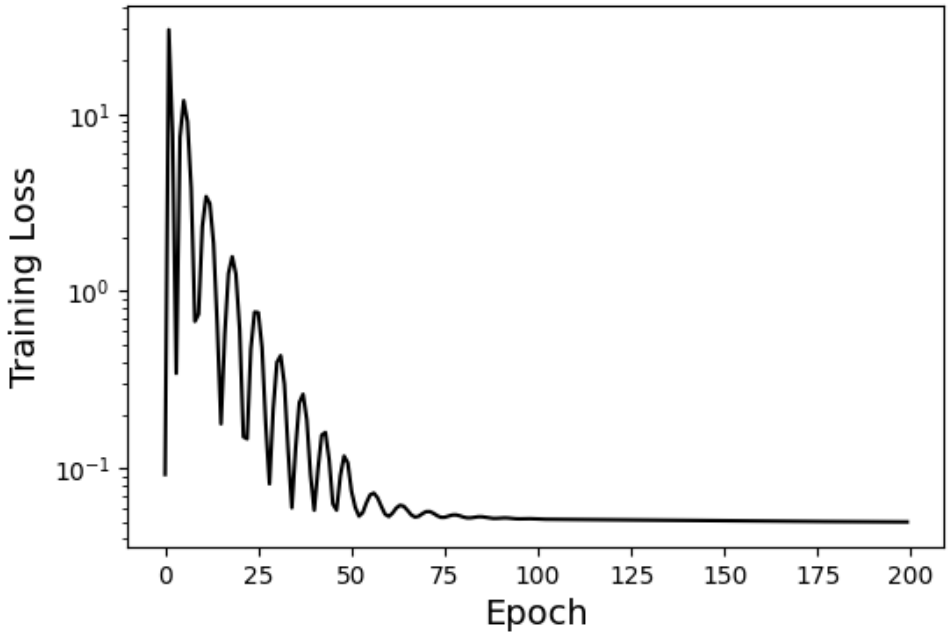

Figure 14. Loss function with $x_{\max} = 2$.

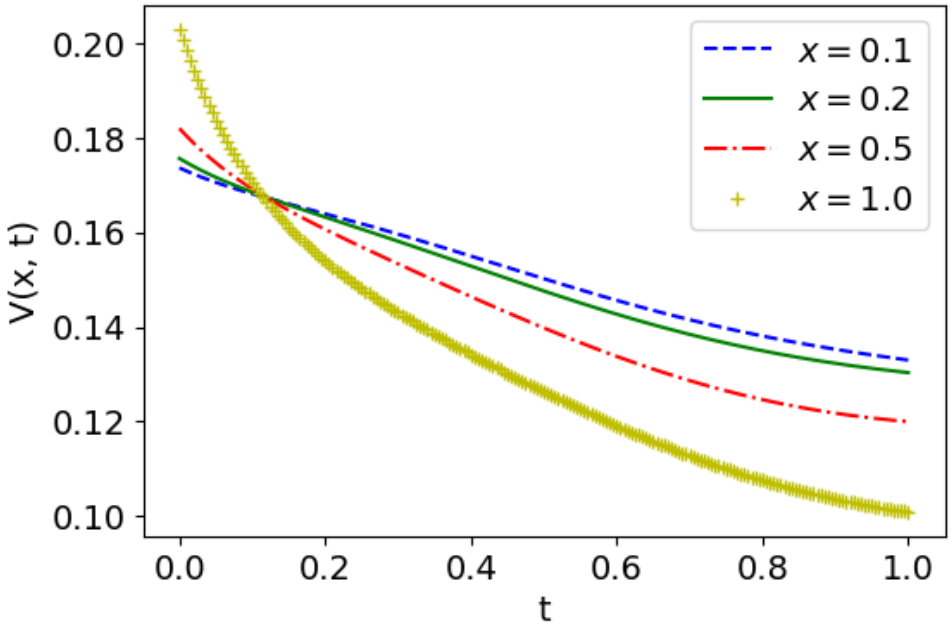

Figure 15. $u(x,t)$ with $x_{\max} = 2$ for specific x.

term. Starting from the one dimension case, once we give a small change ϵ into the corresponding coefficient $a \to a + \epsilon$ (for simplicity, we set $\theta = 0$) from initial equation

$$u_t + au_x^2 - bu_x + \frac{\sigma^2}{2}u_{xx} - h_1 x^2 + c = 0,$$

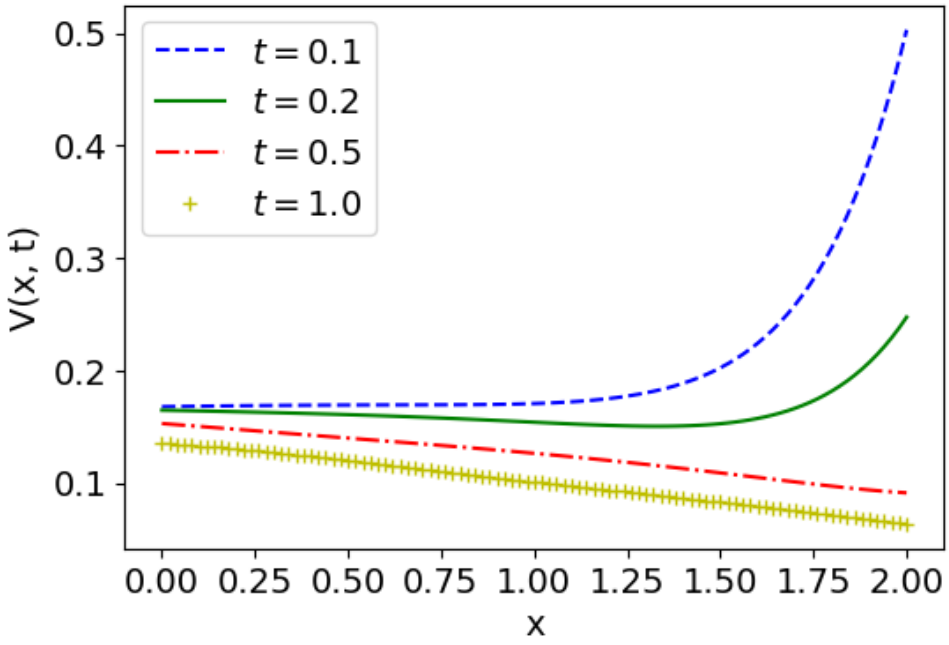

Figure 16. $u(x,t)$ with $x_{\max} = 2$ for fixed t.

to

$$\tilde{u}_t + (a+\epsilon)\tilde{u}_x^2 - b\tilde{u}_x + \frac{\sigma^2}{2}\tilde{u}_{xx} - h_1 x^2 + c = 0,$$

where $\tilde{u} \neq u$ is the new solution and it has the Taylor expansion with respect to ϵ:

$$\tilde{u} = u + u_1\epsilon + u_2\epsilon^2 + ...$$

Now substitute it into the equation and we have:

$$\begin{aligned} &u_t + \epsilon(u_1)_t + \epsilon^2(u_2)_t + ... + (a+\epsilon)\left(u_x + \epsilon(u_1)_x + \epsilon^2(u_2)_x + ...\right)^2 \\ &- b\left(u_x + \epsilon(u_1)_x + \epsilon^2(u_2)_x + ...\right) + \frac{\sigma^2}{2}\left(u_{xx} + \epsilon(u_1)_{xx} + \epsilon^2(u_2)_{xx} + ...\right) - h_1 x^2 + c = 0. \end{aligned} \tag{5.1}$$

Compare with the equation of u and organize it with respect to the different orders of ϵ:

$$\begin{cases} (u_1)_t + 2u_x(u_1)_x + (u_x)^2 - b(u_1)_x + \dfrac{\sigma^2}{2}(u_1)_{xx} = 0, \\ (u_2)_t + (u_1)_x^2 + 2u_x(u_2)_x + 2u_x(u_1)_x - b(u_2)_x + \dfrac{\sigma^2}{2}(u_2)_{xx} = 0, \\ .... \end{cases} \tag{5.2}$$

To solve u_1 once we have already known value of u, there exists no nonlinear term:

$$(u_1)_t + a(u_1)_x + \frac{\sigma^2}{2}(u_1)_{xx} + b = 0.$$

We may apply operator splitting method to solve this. Operator splitting [29] separates the original equation into two parts based on a small time step Δt. We can pick this step as a small

positive number if necessary. In all cases, the computational advantage is that it is faster to compute the solution of the split terms separately, than to compute the solution directly when they are treated together. The classical example is given as following:

$$\frac{du}{dt} = (\mathbf{A} + \mathbf{B})u,$$

which is the ordinary differential equation, here $\mathbf{A}$ and $\mathbf{B}$ is operator for short: ∂_x, ∂_{xx} is such examples. So from the background of Ordinary Differential Equation knowledge, the solution is obvious:

$$u(h) = e^{h(\mathbf{A}+\mathbf{B})}u(0).$$

It is often happening that calculation is relatively difficult to compute, we will compute $e^{h\mathbf{A}}$ and $e^{h\mathbf{B}}$ separately. From the theory, first order splitting is:

$$e^{h(\mathbf{A}+\mathbf{B})} \approx e^{h\mathbf{A}}e^{h\mathbf{B}}.$$

Estimate the error:

$$\begin{aligned} e^{h(\mathbf{A}+\mathbf{B})} &= 1 + h(\mathbf{A}+\mathbf{B}) + \frac{1}{2}h^2(\mathbf{A}+\mathbf{B})^2 + O(h^3) \\ &= 1 + h(\mathbf{A}+\mathbf{B}) + \frac{1}{2}h^2(\mathbf{A}^2 + \mathbf{B}^2 + \mathbf{AB} + \mathbf{BA}), \end{aligned} \tag{5.3}$$

and combine $e^{h\mathbf{A}}$ and $e^{h\mathbf{B}}$ together:

$$\begin{aligned} e^{h\mathbf{A}}e^{h\mathbf{B}} &= (1 + h\mathbf{A} + \frac{1}{2}h^2\mathbf{A}^2 + O(h^3))(1 + h\mathbf{B} + \frac{1}{2}h^2\mathbf{B}^2 + O(h^3)) \\ &= 1 + h\mathbf{A} + h\mathbf{B} + \frac{1}{2}h^2(\mathbf{A}^2 + \mathbf{B}^2) + h^2\mathbf{AB} + O(h^3) \\ &= 1 + h(\mathbf{A}+\mathbf{B}) + \frac{1}{2}h^2(\mathbf{A}^2 + \mathbf{B}^2 + 2\mathbf{AB}) + O(h^3). \end{aligned} \tag{5.4}$$

Then $e^{h(\mathbf{A}+\mathbf{B})} - e^{h\mathbf{A}}e^{h\mathbf{B}} = O(h^2)$ since $\mathbf{AB} - \mathbf{BA} \neq 0$. The second order splitting (higher order) is:

$$e^{h(\mathbf{A}+\mathbf{B})} \approx e^{\frac{1}{2}h\mathbf{A}}e^{h\mathbf{B}}e^{\frac{1}{2}h\mathbf{A}}.$$

Estimate the error:

$$\begin{aligned} e^{h(\mathbf{A}+\mathbf{B})} &= 1 + h(\mathbf{A}+\mathbf{B}) + \frac{1}{2}h^2(\mathbf{A}+\mathbf{B})^2 + O(h^3) \\ &= 1 + h(\mathbf{A}+\mathbf{B}) + \frac{1}{2}h^2(\mathbf{A}^2 + \mathbf{B}^2 + \mathbf{AB} + \mathbf{BA}) + O(h^3), \end{aligned} \tag{5.5}$$

$$\begin{aligned} e^{\frac{1}{2}h\mathbf{A}}e^{h\mathbf{B}}e^{\frac{1}{2}h\mathbf{A}} &= (1 + \frac{1}{2}h\mathbf{A} + \frac{1}{4}h^2\mathbf{A}^2)(1 + \frac{1}{2}h\mathbf{B} + \frac{1}{4}h^2\mathbf{B}^2)(1 + \frac{1}{2}h\mathbf{A} + \frac{1}{4}h^2\mathbf{A}^2) \\ &= 1 + h\mathbf{A} + h\mathbf{B} + \frac{1}{2}h^2\mathbf{A}^2 + \frac{1}{2}h^2\mathbf{B}^2 + \frac{1}{2}h^2\mathbf{AB} + \frac{1}{2}h^2\mathbf{BA} + O(h^3). \end{aligned} \tag{5.6}$$

Then $e^{h(\mathbf{A}+\mathbf{B})} - e^{\frac{1}{2}h\mathbf{A}}e^{h\mathbf{B}}e^{\frac{1}{2}h\mathbf{A}} = O(h^3)$. If the operators commute, the approximation is exact. That is, if the commutator $[\mathbf{A}, \mathbf{B}] \equiv \mathbf{AB} - \mathbf{BA} = 0$, then $e^{h(\mathbf{A}+\mathbf{B})} = e^{h\mathbf{A}}e^{h\mathbf{B}}$. If $\mathbf{AB} = \mathbf{BA}$, then " $\approx$ " can change to " = ". We only consider the first-order method in this paper and higher

order results can be derived similarly through that formula if necessary. Also we can extend the model into three operators $\mathbf{A}, \mathbf{B}, \mathbf{C}$. In our example, we divide the equation into two parts:

$$\begin{cases} (u_1)_t = -a(u_1)_x - b = \mathcal{L}_1(u_1), \\ (u_1)_t = -\dfrac{\sigma^2}{2}(u_1)_{xx} = \mathcal{L}_2(u_1), \end{cases} \tag{5.7}$$

where two operators $\mathcal{L}_1, \mathcal{L}_2$ do not commute. First subpart can be solved through method of characteristic and second one is the classical heat equation. Once we have the solutions of u, u_1, the higher order of $u_i (i \geq 2)$ can be solved with iteration. We omit the details for the length of the paper.

Remark 5.1. The change on other coefficients will be discussed like previous analysis. Normally the government or the central institution make a policy perturbation, then the new optimal problem solution can be derived through this "PDE" method. Another common method applied in the economics is Gateaux derivative in the direction of new objective function τ: The tax reform examples in [28] and [5].

$$\lim_{\mu \to 0} \frac{1}{\mu}[l(x; T + \mu\tau) - l(x; T)].$$

Generally, these two methods are same from mathematical view. The advantage of "PDE" method is clearly finding the optimal solution. However, solving the complicated PDE numerically is not an easy job. Also the "Gateaux derivative" from first order condition is a quick way to find the relation between different coefficients and functions. But the corresponding relation can not help us to determine the optimal value exactly.

6. Conclusion

Our contributions are mainly two-fold. Firstly, we analyze the nonlinear PDE deriving from corresponding HJB equation analytically following the previous work of guessing the solution form exactly. The result shows several disadvantages: (1) We can not guarantee the existence and uniqueness of the solution. (2) The specific solution form can not express all the possibilities. Also the classical finite difference and finite element methods can be applied to solve the PDE model. The refined mesh is necessary in such method which increases the computing time. For the high dimension case, the general numerical method is not efficient any more.

Instead, we follow the PINN method proposed in [26] for solving our model equation with nonlinear term. The advantages of such method has at least three aspects: (1) We do not have constraints about the initial values of the equation since "guessing solution" method depends on the function structure. (2) For one dimension equation case, we have multiple methods, including analytical result, numerical simulation. But once we add more factor variables, the model changes to 2D, 3D or even higher dimensions. Then the tradition idea does not work. Deep learning method is efficient to deal with this difficulty. The training of PINNs does not require discretization of the temporal or spatial domains. At the same time, their practical usage avoid a large number of iterative computations necessary for numerical approaches such as finite element method (FEM). The drawback exists that we have no idea about the properties of the solution theoretically. It can be extended to solve other similar model equations proposed in operation research which is a good future research path for us.

Declaration

All authors contributed to the study conception and design. Material preparation and analysis were performed by Shijing Si. The first draft of the manuscript was written by Yijin Gao and all authors commented on previous versions of the manuscript. The authors have no relevant financial or non-financial interests to disclose.

References

[1] A. G. Baydin, B. A. Pearlmutter, A. A. Radul and J. M. Siskind, *Automatic differentiation in machine learning: A survey*, Journal of Marchine Learning Research, 2018, 18, 1–43.

[2] A. Bihlo, *Improving physics-informed neural networks with meta-learned optimization*, Journal of Machine Learning Research, 2024, 25(14), 1–26.

[3] H.-J. Bungartz, A. Heinecke, D. Pflüger and S. Schraufstetter, *Option pricing with a direct adaptive sparse grid approach*, Journal of Computational and Applied Mathematics, 2012, 236(15), 3741–3750.

[4] Q. Cao, S. Goswami and G. E. Karniadakis, *Laplace neural operator for solving differential equations*, Nature Machine Intelligence, 2024, 6(6), 631–640.

[5] Y. Chang and Y. Park, *Optimal taxation with private insurance*, The Review of Economic Studies, 2021, 88(6), 2766–2798.

[6] L. Feng and X. Lin, *Pricing bermudan options in Lévy process models*, SIAM Journal on Financial Mathematics, 2013, 4(1), 474–493.

[7] Y. Gao, *Operator splitting method for the stochastic production-inventory model equation*, Computers & Industrial Engineering, 2022, 108712.

[8] K. He, X. Zhang, S. Ren and J. Sun, *Deep residual learning for image recognition*, in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, 770–778.

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

[10] Z. Hu, K. Shukla, G. E. Karniadakis and K. Kawaguchi, *Tackling the curse of dimensionality with physics-informed neural networks*, Neural Networks, 2024, 176, 106369.

[11] S. Huang, W. Feng, C. Tang, et al., *Partial differential equations meet deep neural networks: A survey*, IEEE Transactions on Neural Networks and Learning Systems, 2025, 36(8), 13649–13669.

[12] A. Iftakher, R. Golder, B. N. Roy and M. F. Hasan, *Physics-informed neural networks with hard nonlinear equality and inequality constraints*, Computers & Chemical Engineering, 2025, 109418.

[13] M. K. Kadalbajoo, L. P. Tripathi and A. Kumar, *Second order accurate imex methods for option pricing under merton and kou jump-diffusion models*, Journal of Scientific Computing, 2015, 65(3), 979–1024.

[14] J. D. M.-W. C. Kenton and L. K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, in Proceedings of NAACL-HLT, 2019, 4171–4186.

[15] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint. arXiv:1412.6980, 2014.

[16] E. Kiyani, K. Shukla, J. F. Urbán, et al., *Optimizing the optimizer for physics-informed neural networks and kolmogorov-arnold networks*, Computer Methods in Applied Mechanics and Engineering, 2025, 446, 118308.

[17] Y. Kwon and Y. Lee, *A second-order tridiagonal method for american options under jump-diffusion models*, SIAM Journal on Scientific Computing, 2011, 33(4), 1860–1872.

[18] I. E. Lagaris, A. Likas and D. I. Fotiadis, *Artificial neural networks for solving ordinary and partial differential equations*, IEEE Trans. Neural Networks, 1998, 9(5), 987–1000.

[19] Y. LeCun, Y. Bengio and G. Hinton, *Deep learning*, Nature, 2015, 521(7553), 436–444.

[20] S. Li, J. Zhang and W. Tang, *Joint dynamic pricing and inventory control policy for a stochastic inventory system with perishable products*, International Journal of Production Research, 2015, 53(10), 2937–2950.

[21] L. Lu, X. Meng, Z. Mao and G. E. Karniadakis, *Deepxde: A deep learning library for solving differential equations*, SIAM Review, 2021, 63(1), 208–228.

[22] Y. Lu and J. Lu, *A universal approximation theorem of deep neural networks for expressing probability distributions*, Advances in Neural Information Processing Systems, 2020, 33, 3094–3105.

[23] F. Naderi, I. Perez-Raya, S. Yadav, et al., *Towards chemical source tracking and characterization using physics-informed neural networks*, Atmospheric Environment, 2024, 334, 120679.

[24] K. Parand, A. A. Aghaei, S. Kiani, et al., *A neural network approach for solving nonlinear differential equations of lane–emden type*, Engineering with Computers, 2024, 40(2), 953–969.

[25] A. Paszke, S. Gross, F. Massa, et al., *Pytorch: An imperative style, high-performance deep learning library*, Advances in Neural Information Processing Systems, 2019, 32.

[26] M. Raissi, P. Perdikaris and G. 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, 2019, 378, 686–707.

[27] M. Raissi, A. Yazdani and G. E. Karniadakis, *Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations*, Science, 2020, 367(6481), 1026–1030.

[28] D. Sachs, A. Tsyvinski and N. Werquin, *Nonlinear tax incidence and optimal taxation in general equilibrium*, Econometrica, 2020, 88(2), 469–493.

[29] G. Strang, *On the construction and comparison of difference schemes*, SIAM Journal on Numerical Analysis, 1968, 5(3), 506–517.

[30] J. Toivanen, *A high-order front-tracking finite difference method for pricing american options under jump-diffusion models*, The Journal of Computational Finance, 2010, 13(3), 61.

[31] W. Wang, Y. Chen and H. Fang, *On the variable two-step IMEX BDF method for parabolic integro-differential equations with nonsmooth initial data arising in finance*, SIAM Journal on Numerical Analysis, 2019, 57(3), 1289–1317.

Received December 2025; Accepted April 2026; Available online May 2026.