symmetry

MDPI

Article

Solution of Ruin Probability for Continuous Time Model Based on Block Trigonometric Exponential Neural Network

Yinghao Chen [1], Chun Yi [2], Xiaoliang Xie [3], Muzhou Hou [1,*] and Yangjin Cheng [4]

[1] School of Mathematics and Statistics, Central South University, Changsha 410083, China; chenyinghao@csu.edu.cn
[2] College of Finance and Statistics, Hunan University, Changsha 410006, China; yichun@hun.edu.cn
[3] School of Mathematics and Statistics, Hunan University of Technology and Business, Changsha 410205, China; 1941@hunc.edu.cn
[4] School of Mathematics and Computational Science, Xiangtan University, Xiangtan 411105, China; yjcheng@xtu.edu.cn
* Correspondence: hmzw@csu.edu.cn; Tel.: +86-137-8708-8322

Received: 28 April 2020; Accepted: 22 May 2020; Published: 26 May 2020

Abstract: The ruin probability is used to determine the overall operating risk of an insurance company. Modeling risks through the characteristics of the historical data of an insurance business, such as premium income, dividends and reinvestments, can usually produce an integral differential equation that is satisfied by the ruin probability. However, the distribution function of the claim inter-arrival times is more complicated, which makes it difficult to find an analytical solution of the ruin probability. Therefore, based on the principles of artificial intelligence and machine learning, we propose a novel numerical method for solving the ruin probability equation. The initial asset u is used as the input vector and the ruin probability as the only output. A trigonometric exponential function is proposed as the projection mapping in the hidden layer, then a block trigonometric exponential neural network (BTENN) model with a symmetrical structure is established. Trial solution is set to meet the initial value condition, simultaneously, connection weights are optimized by solving a linear system using the extreme learning machine (ELM) algorithm. Three numerical experiments were carried out by Python. The results show that the BTENN model can obtain the approximate solution of the ruin probability under the classical risk model and the Erlang(2) risk model at any time point. Comparing with existing methods such as Legendre neural networks (LNN) and trigonometric neural networks (TNN), the proposed BTENN model has a higher stability and lower deviation, which proves that it is feasible and superior to use a BTENN model to estimate the ruin probability.

Keywords: ruin probability; block trigonometric exponential function; neural network; numerical solution

1. Introduction

In order to help policyholders avoid risks, insurance companies bear all losses for risky buyers [1], which makes insurance companies accumulate a lot of risks themselves [2]. Insurance companies will face austere challenges and tribulations when large claims or a large number of small claims occur in a short period of time [3]. In fact, policyholders focus on the solvency of insurance companies [4]. The ruin probability refers to the probability that insurance business income will be lost at a certain moment, and it is an important indicator to measure the operating status of an insurance company [5]. Therefore, the study of the ruin probability has always been the most active subject in the study of risk theory, especially in ruin theory. Accurate calculations of the ruin probability can help insurance

companies to effectively manage and avoid business risks strategically [6], which has important practical significance.

In the past few decades, researchers have established different types of insurance risk models based on premium income [7,8], payment amount [9], investment [10,11], dividend [12] and other characteristics [13]. Assuming that the distribution of claim intervals is affected by the Poisson process, Lundberg proposed a classic bankruptcy model in 1903 [14]. Sparre Anderson extended the distribution of compensation time intervals from the Poisson process to a general distribution, and constructed an updated risk model in 1957 [15]. Dickson and Hipp calculated the probability of bankruptcy under the Erlang(2) risk model [16]. Subsequently, Li et al. studied the Erlang (n) risk model [17,18]. Considering the uncertainty of the dividend strategy and investment, Gerber proposed a classical risk model with disturbances in 1970 [19]. A Markov process is used to describe the uncertainty of the external environment, and some risk models with Markov parameters were proposed [20–22]. Asmussen summarized different types of bankruptcy risks [23]. Modeling through probability theory and stochastic processes, generally, the ruin probability can be expressed in the form of an integral differential equation [24]. Cai et al. studied the probability of ruin under different interest rates and gave numerical solutions [25], extending the claim time to the generalized Erlang (n) distribution. Since the generalized Lundberg equation may have multiple positive roots, Bergel corresponds the obtained roots to the solutions of the differential equations one-by-one, taking into account factors such as dividends [26]. Kasumo used a block-by-block method to calculate the bankruptcy probability including the investment returns of the standard Black-Scholes model, illustrating the sensitivity of the ruin probability to stock price fluctuations [27].

Due to the complex structure of integral differential equations, it is difficult to use elementary functions to describe the ruin probability, and even these equations often cannot find any analytical solutions. Traditional methods use some inequalities to estimate the range of the ruin probability [28–30]. With the development of numerical calculation techniques such as the Euler method [31,32] and Runge–Kutta method [33], Cardosoa and Waters discussed the numerical solution of the ruin probability in the case of a finite time domain [34]. Makroglou used the polynomial spline method to solve the first-order Volterra integral–differential equations and obtained the approximate solution of the collective non-ruin equation [35]. Paulsen analyzed the probability of ruin with random returns through the block Simpson rule [36]. Tsitsiashvili used an exponential distribution to approximate the loss, and then calculates the ruin probability under the classical risk model with a claims distribution [37]. Zhang proposed the use of a Fourier cosine expansion to estimate the ruin probability in the compound Poisson distribution risk model, and estimated the error of the numerical solution [38].

As a powerful machine learning method, the neural network is widely used in computer vision [39,40], natural environment simulation [41–43], price prediction [44–46] and numerical solutions of differential equations [47–51]. Sabir proposed the use of Morlet wavelet neural networks for solving second-order differential equation [52]. Hure at al. proposed a deep backward neural network for solving high-dimension PDEs [53]. Samaniego uses deep neural networks in the partial differential equations of computational mechanics to improve the computational efficiency of mechanical problems [54]. In addition, Huang proposed an extreme learning machine (ELM) method for fast training a single hidden layer feedforward neural network [55]. In 2019, Zhou and Hou proposed to use the improved ELM (IELM) method and trigonometric basis function neural networks (TNN) to numerically solve the ruin probability [56]. Defining Legendre polynomials as the activation function in the hidden layer, Legendre neural network (LNN) based on IELM was used by Lu to obtain the approximate solution of the ruin probability under the Erlang(2) risk model and the classical risk model [57].

In this study, we apply the block trigonometric exponential neural network (BTENN) to find the approximate solutions of the ruin probability in the classical risk model and the Erlang(2) risk model separately. Numerical experiments were programmed in Python 3.6 and ran on a MacBook Pro (Retina,

13-inch, Late 2013). Comparing the numerical solutions obtained by the proposed BTENN method with LNN and TNN methods, BTENN method has the advantages of high-precision and stability.

The rest of this study is structured as follows: Section 2 briefly reviews the ELM algorithm. The block trigonometric exponential neural network (BTENN) is proposed in Section 3. Section 4 discusses the methods and details of using the BTENN model to obtain the numerical solution of the ruin probability equation in the case of the classical risk model and the Erlang(2) risk model. Three numerical experiments are conducted in Section 5, which proves the accuracy and reliability of the BTENN model. Finally, conclusions and prospects are given in Section 6.

2. Extreme Learning Machine Algorithm

As shown in Figure 1. The ELM is an algorithm that is used to optimize the parameters in a single hidden layer feedforward neural network (SLFN). Since it does not require repeated iterations in the calculation process, only the Moore–Penrose generalized inverse of the coefficient matrix is needed to obtain the global optimal solution. Thus, it has the advantages of a fast calculation speed and does not easily overfit. For a given data set with M samples $\{(x_i, y_i)\}_{i=1}^{M} \in \mathbb{R}^{M\times(N+1)}$, where N is the number of features, The output of this single hidden layer network with m hidden neurons can be mathematical represented as follows:

$$f_m(x) = \sum_{i=1}^{m} \beta_i G(a_i, b_i, x) = h(x)\beta = \widetilde{y}, \tag{1}$$

where $h(x) = [G(a_1, b_1, x), G(a_2, b_2, x), \cdots, G(a_m, b_m, x)]$ is the group of activation function which projects the network input x into the m-dimensional feature space. In addition, $\beta = [\beta_1, \beta_2, \cdots, \beta_m]^T$ is the connection weight between hidden layer and output layer. When the network approximates the given data set without error by ELM algorithm, this means Equation (1) satisfies each data in the given dataset, which can be rewritten as:

$$f_m(x_j) = \sum_{i=1}^{m} \beta_i G(a_i, b_i, x_j) = h(x_j)\beta = y_j \text{ j} = 1, 2, \cdots \text{M}, \tag{2}$$

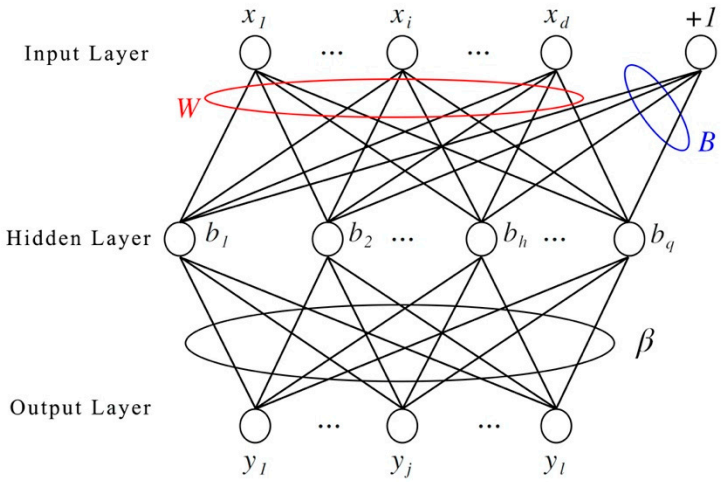

Figure 1. The topology structure of the extreme learning machine (ELM) algorithm.

Equation (2) can be expressed in matrix form, i.e.,

$$H\beta = y, \tag{3}$$

where,

$$H = \begin{bmatrix} h(x_1) \\ \vdots \\ h(x_M) \end{bmatrix} = \begin{bmatrix} h_1(x_1) & \cdots & h_m(x_1) \\ \vdots & \ddots & \vdots \\ h_1(x_M) & \cdots & h_m(x_M) \end{bmatrix}, y = \begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix}, \tag{4}$$

Then the problem of finding the optimal connection weight is transformed into the following optimization problem:

$$\min : \sum_{i=1}^{M} \xi_i^2$$
$$\text{s.t. } \mathrm{h}(x_i)\beta = y_i - \xi_i$$

Generally, the number of training points M will much bigger than the number of features. Based on the Karush–Kuhn–Tucker (KKT) condition, For any input weight a_i and biases b_i, ELM algorithm can get the minimum square error of the output through the least square method and it can be represented as

$$\beta^* = \underset{\beta}{argmin} \|H\beta - y\|^2 = H^{\dagger} y, \tag{5}$$

where, $\mathrm{H}^{\dagger} = \left(\mathrm{H}^{\mathrm{T}}\mathrm{H}\right)^{-1}\mathrm{H}^{\mathrm{T}}$ is the Moore–Penrose generalized inverse of matrix H.

The steps of the ELM algorithm are composed of three parts:

Step 1. Randomly initialized parameters in the hidden layer;
Step 2. Calculate the output matrix of the hidden layer;
Step 3. Obtain the output weight β by least square method.

3. Block Trigonometric Exponential Neural Network

In this study, a single hidden layer feedforward neural network (SLFN) model is established. The block trigonometric exponential function neural network (BTENN) is a special case of SLFN, which has an input node x and one output layer. We propose a novel trigonometric exponential expansion to replace the activation function of the hidden layer (generally a Gaussian function or the Legendre polynomials). The unique hidden layer consists of two parts. The first part uses the cosine exponential function as the basis function, and the other part implements the superposition of the sine exponential function. The input vector x is projected into a high-dimensional feature space using the proposed trigonometric exponential basis function, where cosine exponential basis functions and sine exponential basis functions are denoted as $G_c(x)$ and $G_s(x)$, respectively. Two sets of basis functions are symmetrical to each other. These nonlinear basis functions ensure that the BTENN has the ability to fit complex ruin probabilities. The general form of the trigonometric exponential basis function can be written as follows:

$$G_{i,c}(x) = (1 - cosx)^2 e^{-\frac{k}{i}x}, \tag{6}$$

$$G_{i,s}(x) = (1 - sinx)^2 e^{-\frac{k}{i}x}, \tag{7}$$

Consider an input vector $x = (x_1, x_2, \cdots, x_m)$, of whose dimension is m. The coordinates projected by trigonometric exponential functions in a hidden layer can be calculated as:

$$\begin{bmatrix} \mathrm{G}_{1,\mathrm{c}}(\mathrm{x}_1) & \cdots & \mathrm{G}_{\frac{\mathrm{N}}{2},\mathrm{c}}(\mathrm{x}_1) & \mathrm{G}_{1,\mathrm{s}}(\mathrm{x}_1) & \cdots & \mathrm{G}_{\frac{\mathrm{N}}{2},\mathrm{s}}(\mathrm{x}_1) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathrm{G}_{1,\mathrm{c}}(\mathrm{x}_\mathrm{m}) & \cdots & \mathrm{G}_{\frac{\mathrm{N}}{2},\mathrm{c}}(\mathrm{x}_\mathrm{m}) & \mathrm{G}_{1,\mathrm{s}}(\mathrm{x}_\mathrm{m}) & \cdots & \mathrm{G}_{\frac{\mathrm{N}}{2},\mathrm{s}}(\mathrm{x}_\mathrm{m}) \end{bmatrix}, \tag{8}$$

where N is the number of hidden neurons. In the BTENN, N is always an even number. A schematic diagram of a block trigonometric exponential function neural network is given in Figure 2. Linearly combined with appropriate coefficients α_i, the output of the BTENN can be calculated as:

$$N(x,\ \alpha) = \sum_{i=1}^{\frac{N}{2}} [\alpha_{i,1} G_{i,c}(x) + \alpha_{i,2} G_{i,s}(x)]. \tag{9}$$

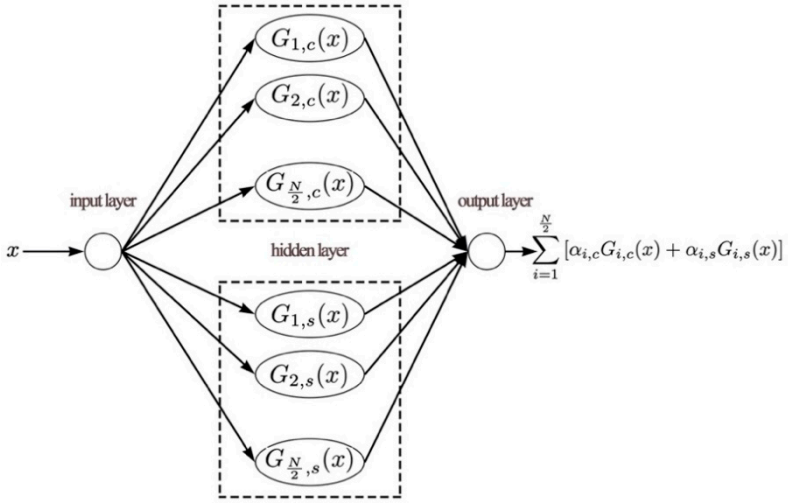

Figure 2. Structure of a block trigonometric exponential neural network.

Remark 1. *Convergence theorem*

We claim that the output of the block trigonometric exponential neural network can converge to any continuous function $\psi(x)$.

Proof. let $L^2(R)$ be a Hilbert space and $G_{n,c}(\mathrm{x}) = (1 - \cos \mathrm{x})^2 e^{-\frac{kx}{n}}$, $G_{n,s}(\mathrm{x}) = (1 - \sin \mathrm{x})^2 e^{-\frac{kx}{n}}$. where, $G_{n,c}(\mathrm{x})$ and $G_{n,s}(\mathrm{x})$ form a pair basis in $L^2(R)$.

Let $U(\mathrm{x}) = \sum_{i=1}^{N} \langle \alpha_i, G_i(x) \rangle$ be the output of block trigonometric exponential neural networks, where $\langle \cdot, \cdot \rangle$ represents the inner product, and

$$\alpha_i = (\alpha_{i,c},\ \alpha_{i,s}), \tag{10}$$

$$G_i(x) = (G_{i,c}(x), G_{i,s}(x)). \tag{11}$$

□

That is,

$$U(\mathrm{x}) = \sum_{i=1}^{N} \Big\langle (\alpha_{i,c},\ \alpha_{i,s}), (G_{i,c}(x), G_{i,s}(x)) \Big\rangle = \sum_{i=1}^{n} [\alpha_{i,c} \cdot G_{i,c}(x) + \alpha_{i,s} \cdot G_{i,s}(x)]. \tag{12}$$

Here, we denote the sequence of partial sums of $\left\{ \left\langle \alpha_{i,c} + \alpha_{i,s}, G_{i,c}(x) + G_{i,s}(x) \right\rangle \right\}$ as $\left\{ S^i \right\}$.

Hence, S^m and S^n represent the arbitrary partial sums with $m \geq n > 0$, we need to prove that $\left\{ S^i \right\}$ is a Cauchy sequence.

Considering that $S^n = \sum_i^n \langle \alpha_i, G_i(x) \rangle$ and $S^m = \sum_i^m \langle \alpha_i, G_i(x) \rangle$, in this case, we can get:

$$\begin{aligned} \| S^m - S^n \|^2 = \| \sum_{i=1}^{m} \langle \alpha_i, G_i(x) \rangle \|^2 = \| \sum_{i=n}^{m} \alpha_{i,c} \cdot (1 - \cos x)^2 e^{-\frac{kx}{i}} + \alpha_{i,s} \cdot (1 - \sin x)^2 e^{-\frac{kx}{i}} \|^2 \\ \leq \| \sum_{i=n}^{m} 2\alpha_{i,c} \cdot e^{-\frac{kx}{i}} + \alpha_{i,s} \cdot e^{-\frac{kx}{i}} \|^2 \\ \leq \| \sum_{i=n}^{m} 4\alpha_{i,max} \cdot e^{-\frac{kx}{i}} \|^2 \leq \| \sum_{i=n}^{m} 4\alpha_{i,max} \|^2 \\ = 16 \sum_{i=n}^{m} \alpha_{i,max}^2, \end{aligned} \tag{13}$$

where $\alpha_{i,max} = \max\limits_{i=1,2,\cdots,N} \left\{ \left| \alpha_{i,c} \right|, \left| \alpha_{i,s} \right| \right\}$.

As $m, n \to \infty$ in Equation (13), we have $\| S^m - S^n \|^2 \to 0$. This means $\left\{ S^i \right\}$ is a Cauchy sequence in Hilbert space $L^2(R)$ and it converges to:

$$S = \sum\nolimits_i^{\infty} \langle \alpha_i, G_i(x) \rangle. \tag{14}$$

Thus, the BTENN can approximate an unknown continuous function with arbitrary precision, as the number of hidden layer neurons N approaches to infinity.

4. The BTENN Model for the Ruin Probability Equation

In this section, we will introduce the details of solving the ruin probability equation by the block trigonometric exponential neural network under the assumption of the classical risk model or the Erlang(2) risk model.

4.1. Classical Risk Model

The utility function $U(t)$ represents the surplus assets of the insurance company at time t, in the compound risk model, this process can be expressed as,

$$U(t) = u + ct - \sum\nolimits_{i=1}^{N_t} X_i, \ t \geq 0, \tag{15}$$

It has following structures: The sequence of random variables $\{N_t, \ t = 0, 1, 2, \cdots\}$ represents the number of occurrences of the claim during time $(0, \ t]$, which is considered to form a Poisson process with parameter λ. Suppose that the number of claims $\{N_t, \ t = 0, 1, 2, \cdots\}$ and the amount of claims $\{X_i, \ i = 0, 1, \cdots, N_t\}$ are independent of each other. In addition, the amount of the i-th claim X_i is based on a non-negative random sequence with independent identical distribution, whose distribution function is $F(x)$ and mean value is μ, respectively. The initial surplus of an insurance company is $u \geq 0$ and c is the insurance premium charged in a unit time. In order to ensure that the insurance company can operate safely, the premium income should not be less than the expected value of the compensation amount, i.e.,:

$$E\left(ct - \sum\nolimits_{i=0}^{N_t} X_i \right) = (c - \lambda\mu)t \geq 0. \tag{16}$$

This means the security loading $\theta = \frac{c - \lambda\mu}{\lambda\mu} > 0$.

The time of ruin T is defined as:

$$T = \inf\{t \geq 0, U(t) < 0\}, \tag{17}$$

where, as $T = +\infty$ means that the insurance company will never goes bankrupt.

The ultimate ruin probability is expressed as:

$$\psi(u) = P(T < \infty \, | U(0) = u), \ u \geq 0. \tag{18}$$

In other words, the ultimate survival probability $\phi(u)$ is equal to $1-\psi(u)$.

If the surplus process of an insurance company can be described by Formula (15), the ultimate ruin probability satisfies the following integral differential equation and initial value condition:

$$\psi'(u) = \frac{\lambda}{c}\psi(u) - \frac{\lambda}{c}\int_0^u \psi(u-x)dF(x) - \frac{\lambda}{c}[1-F(u)], \tag{19}$$

$$\psi(0) = \frac{\lambda}{c}\mu = \frac{1}{1+\theta}, \text{ when } c > \lambda\mu. \tag{20}$$

4.2. BTENN for Classical Risk Model

According to the BTENN model proposed above, the trial solution can be constructed as follows:

$$\widetilde{\varphi}(u) = \frac{1}{1+\theta} + u \cdot \sum_{i=1}^{N/2}[\alpha_{i,c} \cdot G_{i,c}(u) + \alpha_{i,s} \cdot G_{i,s}(u)] \tag{21}$$

In fact, the trial solution satisfies the initial conditions (20). Replace $\psi(u)$ by the trial solution $\widetilde{\varphi}(u)$ in Equation (19),

$$\widetilde{\varphi}'(u) = \frac{\lambda}{c}\widetilde{\varphi}(u) - \frac{\lambda}{c}\int_0^u \widetilde{\varphi}(u-x)dF(x) - \frac{\lambda}{c}[1-F(u)]. \tag{22}$$

Expanding expression (22), we have:

$$LHS = \widetilde{\varphi}'(u) = \sum\nolimits_{i=1}^{N/2}\Big[\alpha_{i,c} \cdot (G_{i,c}(u) + uG_{i,c}(u)' + \alpha_{i,s} \cdot \Big(G_{i,s}(u) + uG_{i,s}(u)'\Big)\Big], \tag{23}$$

$$RHS = RHS_1 - RHS_2 - RHS_3, \tag{24}$$

where,

$$\begin{aligned} RHS_1 &= \frac{\lambda}{c}\widetilde{\varphi}(u) = \frac{\lambda}{c}[\frac{1}{1+\theta} + \sum_{i=1}^{\frac{N}{2}} u[\alpha_{i,c}G_{i,c}(u) + \alpha_{i,s}G_{i,s}(u)] \\ &= \frac{\lambda}{c(1+\theta)} + \sum_{i=1}^{\frac{N}{2}}\Big[\alpha_{i,c}\frac{\lambda}{c}uG_{i,c}(u) + \alpha_{i,s}\frac{\lambda}{c}uG_{i,s}(u)\Big] \end{aligned} \tag{25}$$

$$\begin{aligned} RHS_2 &= \tfrac{\lambda}{c}\int_0^u \widetilde{\varphi}(u-x)dF(x) \\ &= \tfrac{\lambda}{c}\int_0^u \tfrac{1}{1+\theta} + (u-x)\sum_{i=1}^{\frac{N}{2}} u[\alpha_{i,c}G_{i,c}(u-x) + \alpha_{i,s}G_{i,s}(u-x)]dF(x) \\ &= \tfrac{\lambda}{c}\tfrac{1}{1+\theta}[F(u)-F(0)] + \sum_{i=1}^{\frac{N}{2}} \alpha_{i,c}\tfrac{\lambda}{c}\int_0^u (u-x)G_{i,c}(u-x)dF(x) \\ &\quad + \textstyle\sum_{i=1}^{\frac{N}{2}} \alpha_{i,s}\tfrac{\lambda}{c}\int_0^u (u-x)G_{i,s}(u-x)dF(x), \end{aligned} \tag{26}$$

and

$$RHS_3 = \frac{\lambda}{c}[1-F(u)]. \tag{27}$$

Merging the similar items, Equation (22) can be rewritten as,

$$\begin{aligned} &\textstyle\sum_{i=1}^{\frac{N}{2}}\Big[G_{i,c}(u) - \tfrac{\lambda}{c}uG_{i,c}(u) + uG'_{i,c}(u) + \tfrac{\lambda}{c}\int_0^u (u-x)G_{i,c}(u-x)dF(x)\Big] \\ &+ \textstyle\sum_{i=1}^{\frac{N}{2}}\Bigg[G_{i,s}(u) - \tfrac{\lambda}{c}uG_{i,s}(u) + uG'_{i,s}(u) + \tfrac{\lambda}{c}\int_0^u (u-x)G_{i,s}(u-x)dF(x)\Bigg] \\ &= \tfrac{\lambda}{c}\Big[\tfrac{F(\theta)-F(u)-\theta}{1+\theta} + F(u)\Big]. \end{aligned} \tag{28}$$

Take a series of training points $\Omega = 0 = u_1 < u_2 \ldots < u_m = b$, one clear fact is that Equation (28) holds at these training points, that is,

$$\begin{aligned}&\sum_{i=1}^{\frac{N}{2}}\left[\left(1-\tfrac{\lambda}{c}u_j\right)G_{i,c}\left(u_j\right)+u_jG'_{i,c}\left(u_j\right)+\tfrac{\lambda}{c}\int_0^{u_j}\left(u_j-x\right)G_{i,c}\left(u_j-j\right)dF(x)\right]\\&+\sum_{i=1}^{\frac{N}{2}}\left[\left(1-\tfrac{\lambda}{c}u_j\right)G_{i,s}\left(u_j\right)+u_jG'_{i,s}\left(u_j\right)+\tfrac{\lambda}{c}\int_0^{u_j}\left(u_j-x\right)G_{i,s}\left(u_j-j\right)dF(x)\right]\\&=\tfrac{\lambda}{c}\left[\tfrac{F(\theta)-F\left(u_j\right)-\theta}{1+\theta}+F\left(u_j\right)\right],\ \ j=1,2,\cdots,M,\end{aligned}\tag{29}$$

Finally, the linear system in (29) has a matrix form,

$$A\alpha = B,\tag{30}$$

where,

$$A=\left(\ AC,AS\ \right)_{M\times N},\ AC=\begin{pmatrix}AC_1\\AC_2\\\vdots\\AC_M\end{pmatrix},\ AS=\begin{pmatrix}AS_1\\AS_2\\\vdots\\AS_M\end{pmatrix},\tag{31}$$

$$AC_{ij}=\left(1-\frac{\lambda}{c}u_j\right)\cdot G_{i,c}\left(u_j\right)+u_jG'_{i,c}\left(u_j\right)+\frac{\lambda}{c}\int_0^{u_j}\left(u_j-x\right)G_{i,c}\left(u_j-x\right)dF(x),\tag{32}$$

$$AS_{ij}=\left(1-\frac{\lambda}{c}u_j\right)\cdot G_{i,s}\left(u_j\right)+u_jG'_{i,s}\left(u_j\right)+\frac{\lambda}{c}\int_0^{u_j}\left(u_j-x\right)G_{i,s}\left(u_j-x\right)dF(x)\tag{33}$$

and,

$$\alpha^T=\left(\ \alpha_c,\alpha_s\ \right)_{1\times N},\ \alpha_c=\begin{pmatrix}\alpha_{1,\,c}\\\alpha_{2,\,c}\\\vdots\\\alpha_{\frac{N}{2},\,c}\end{pmatrix},\ \alpha_s=\begin{pmatrix}\alpha_{1,\,s}\\\alpha_{2,\,s}\\\vdots\\\alpha_{\frac{N}{2},\,s}\end{pmatrix},\tag{34}$$

$$B=\left(\tfrac{\lambda}{c}\cdot\tfrac{F(0)-F(u_0-\theta)}{1+\theta}+F(u_0),\ \ \tfrac{\lambda}{c}\cdot\tfrac{F(0)-F(u_1-\theta)}{1+\theta}+F(u_1),\ \cdots,\tfrac{\lambda}{c}\cdot\tfrac{F(0)-F(u_m-\theta)}{1+\theta}+F(u_m)\right)^T.\tag{35}$$

Note that, in generally, the original function of the definite integrals $\int_0^{u_j}\left(u_j-x\right)G_{i,c}(u)\left(u_j-x\right)dF(x)$ and $\int_0^{u_j}\left(u_j-x\right)G_{i,s}(u)\left(u_j-x\right)dF(x)$ in Formulas (32) and (33) cannot be represented by elementary functions. We usually use its numerical integration by Simpson's formula [58]:

$$\begin{aligned}\int_a^b f(x)dx\approx S_n&=\frac{h}{b}\sum_{k=0}^{n-1}\left[f(x_k)+4f\left(x_{k+\frac{1}{2}}\right)+f(x_{k+1})\right]\\&=\frac{h}{b}\left[f(a)+4\sum_{k=0}^{n-1}f\left(x_{k+\frac{1}{2}}\right)+2\sum_{k=1}^{n}f(x_k)+f(b)\right].\end{aligned}\tag{36}$$

The optimal coefficient vector α by ELM method is,

$$\alpha^* = \mathrm{A}^{\dagger}\mathrm{B},\tag{37}$$

where, $\mathrm{A}^{\dagger}$ is the least square solution to the linear system (30).

4.3. Erlang(2) Risk Model

By generalizing the claim inter-arrival time of classical risk model from the exponential distribution to an Erlang (n) distribution, we can get an Erlang (n) renewal risk model, which may better describe the risks encountered by insurance companies in practical problems. In particular, this study focuses on the Erlang(2) risk model: its surplus process can be expressed as

$$U(n) = u + \sum_{i=1}^{n}(cT_i - X_i), \;\; n \geq 1, \tag{38}$$

where u represents the initial reserve of the insurance company, n denotes the number of claims, c is the insurance premium charged per unit time and T_i is the time interval between the $i-$ th and $(i-1)-th$ payments. As in the classical risk model, we assume that T_i is independent of the claim amount X_i. Moreover, suppose that the time interval T_i is a non-negative sequence with independent and identical distribution. In the Erlang(2) model, the probability density function of the claim interval is $f(t) = \eta^2 t e^{-\eta t}$.

When the surplus is negative, the insurance company is considered bankrupt. The ruin probability is denoted by $\psi(u)$, whose definition is

$$\psi(u) = P\Big\{u + \sum_{i=1}^{n}(cT_i - X_i) < 0, \; \exists n \in \mathbb{N}\Big\}.$$

The ruin probability of insurance company $\psi(u)$ satisfies the following equation [16]:

$$c^2\psi''(u) - 2\eta c\psi'(u) + \eta^2\psi(u) = \eta^2\int_0^u \psi(u-x)dF(x) + \eta^2[1-F(u)], \tag{39}$$

$$\psi(0) = \frac{c^2 s_0 - 2\eta c + \eta^2\alpha}{c^2 s_0}, \tag{40}$$

where $F(x)$ is the distribution function of claim size X_i and s_0 is the solution of the following equation:

$$c^2s^2 - 2\eta cs + \eta^2 = \eta^2\int_0^{+\infty} e^{-sx}dF(x). \tag{41}$$

4.4. BTENN for Erlang(2) Risk Model

Similar to the classical risk model above, we claim that the trial solution that satisfies with the given initial value condition is as follows:

$$\widetilde{\varphi}(u) = \psi(0) + u\cdot\sum_{i=1}^{N/2}[\alpha_{i,c}\cdot G_{i,c}(u) + \alpha_{i,s}\cdot G_{i,s}(u)]. \tag{42}$$

Substituting model (38) into Equation (36), we have

$$c^2\widetilde{\varphi}''(u) - 2\eta c\widetilde{\varphi}'(u) + \eta^2\widetilde{\varphi}(u) = \eta^2\int_0^u \widetilde{\varphi}(u-x)dF(x) + \eta^2[1-F(u)], \tag{43}$$

That is,

$$LHS = LHS_1 - LHS_2 + LHS_3$$

$$RHS = RHS_1 + RHS_2$$

where,

$$LHS_1 = c^2\widetilde{\varphi}''(u) = c^2\sum_{i=1}^{N/2}[\alpha_{i,c}\cdot(2G'_{i,c}(u) + uG''_{i,c}(u) + \alpha_{i,s}\cdot(2G'_{i,s}(u) + uG''_{i,s}(u)], \tag{44}$$

$$LHS_2 = 2\eta c\widetilde{\varphi}'(u) = 2\eta c\sum\nolimits_{i=1}^{N/2}[\alpha_{i,c}\cdot(G_{i,c}(u)+uG'_{i,c}(u))+\alpha_{i,s}\cdot(G_{i,c}(u)+uG'_{i,s}(u))], \quad (45)$$

$$LHS_3 = \eta^2\widetilde{\varphi}(u) = \eta^2[\psi(0)+u\sum\nolimits_{i=1}^{N/2}[\alpha_{i,c}\cdot G_{i,c}(u)+\alpha_{i,s}\cdot G_{i,s}(u)], \quad (46)$$

$$RHS_1 = \eta^2\int_0^u \widetilde{\varphi}(u-x)dF(x)$$

$$= \eta^2\psi(0)[F(u)-F(0)]+\eta^2\int_0^u(u-x)\sum\nolimits_{i=1}^{N/2}[\alpha_{i,c}\cdot G_{i,c}(u-x)+\alpha_{i,s}\cdot G_{i,s}(u-x)]dF(x), \quad (47)$$

$$RHS_2 = \eta^2[1-F(u)], \quad (48)$$

Sorting out Formulas (44)–(48),

$$\begin{aligned}&\sum\nolimits_{i=1}^{N/2}\alpha_{i,c}\left[c^2\cdot u\cdot G''_{i,c}(u)+2c(c-\eta u)G'_{i,c}(u)+\left(\eta^2u-2\eta c\right)G_{i,c}(u)-\eta^2\int_0^u(u-x)G_{i,c}(u-x)dF(x)\right]\\&+\sum\nolimits_{i=1}^{N/2}\alpha_{i,s}\left[c^2\cdot u\cdot G''_{i,s}(u)+2c(c-\eta u)G'_{i,s}(u)+\left(\eta^2u-2\eta c\right)G_{i,s}(u)-\eta^2\int_0^u(u-x)G_{i,s}(u-x)dF(x)\right]\\&=\eta^2\psi(0)[F(u)-F(0)-1]+\eta^2[1-F(u)],\end{aligned} \quad (49)$$

When Equation (49) holds on the selected training point set $\Omega = 0 = u_1 < u_2 \cdots < um = b$, the above equation can be represented as follows:

$$\begin{aligned}&\sum_{i=1}^{N/2}\alpha_{i,c}\left[c^2\cdot u_j\cdot G''_{i,c}\left(u_j\right)+2c\left(c-\eta u_j\right)G'_{i,c}\left(u_j\right)+\left(\eta^2u_j-2\eta c\right)G_{i,c}\left(u_j\right)-\eta^2\int_0^{u_j}\left(u_j-x\right)G_{i,c}\left(u_j-x\right)dF(x)\right]\\&+\sum\nolimits_{i=1}^{N/2}\alpha_{i,s}\left[c^2\cdot u_j\cdot G''_{i,s}\left(u_j\right)+2c\left(c-\eta u_j\right)G'_{i,s}\left(u_j\right)+\left(\eta^2u_j-2\eta c\right)G_{i,s}\left(u_j\right)-\right.\\&\left.\eta^2\int_0^{u_j}\left(u_j-x\right)G_{i,s}\left(u_j-x\right)dF(x)\right]=\eta^2\psi(0)\left[F\left(u_j\right)-F(0)-1\right]+\eta^2\left[1-F\left(u_j\right)\right],\end{aligned} \quad (50)$$

In matrix form of Equation (50) becomes

$$A\alpha = B$$

where,

$$A = (AC, AS)_{M\times N},\ AC = (AC_1, AC_2, \cdots AC_M)^T,\ AS = (AS_1, AS_2, \cdots AS_M)^T \quad (51)$$

$$AC_{ij} = c^2\cdot u_j\cdot G''_{i,c}\left(u_j\right)+2c\left(c-\eta u_j\right)G'_{i,c}\left(u_j\right)+(\eta^2u_j-2\eta c)G_{i,c}\left(u_j\right)-\eta^2\int_0^{u_j}\left(u_j-x\right)G_{i,c}\left(u_j-x\right)dF(x), \quad (52)$$

$$AS_{ij} = c^2\cdot u_j\cdot G''_{i,s}\left(u_j\right)+2c\left(c-\eta u_j\right)G'_{i,s}\left(u_j\right)+(\eta^2u_j-2\eta c)G_{i,s}\left(u_j\right)-\eta^2\int_0^{u_j}\left(u_j-x\right)G_{i,s}\left(u_j-x\right)dF(x), \quad (53)$$

and

$$\alpha = \begin{pmatrix}\alpha_c\\ \alpha_s\end{pmatrix}_{N\times 1},\ \alpha_c = \left(\alpha_{1,c}, \alpha_{2,c}, \cdots \alpha_{\frac{N}{2},c}\right)^T,\ \alpha_s = \left(\alpha_{1,s}, \alpha_{2,s}, \cdots \alpha_{\frac{N}{2},s}\right)^T$$

$$B_j = \eta^2\psi(0)\left[F\left(u_j\right)-F(0)-1\right]+\eta^2\left[1-F\left(u_j\right)\right],\ j = 1, 2, \cdots, M, \quad (54)$$

Training such a block trigonometric exponential neural network is equivalent to finding the least square solution to a linear system (50). The optimal parameter of the connection weight of the BTENN model can be obtained by the Moore–Penrose generalized inverse of matrix A as

$$\alpha^* = \mathrm{A}^{\dagger}\mathrm{B}$$

Meanwhile, $\alpha^* = \min\limits_{\alpha} \| A\alpha - B \|$ exists and is unique.

5. Numerical Results and Analysis

In this section, three numerical experiments are conducted to verify the efficiency of the proposed BTENN method. In the first and third experiments, we assume that the number of claims follows a compound Poisson distribution, while the second experiment shows the numerical solution of the ruin probability equation when the inter-claim intervals follow an Erlang(2) renewal process. In the classical ELM model, the global optimal solution can be obtained by randomly assigning the weight ω and its bias b in the hidden layer; $\omega_i = 1$ and $b_i = 0$ are set in our experiment for the convenience of calculation.

In order to measure the accuracy and stability of different algorithms, we calculated the mean square error (MSE) of the approximate solutions obtained on the same test set. The MSE can be calculated as follows:

$$MSE = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} (y(x_i) - \widetilde{y}(x_i))^2. \tag{55}$$

In fact, we are not only concerned with mean square error, but also with the maximum absolute error (MAE). In real world problems, MAE determines the stability of the model and whether it can be used in practical applications. MAE is defined as:

$$MAE = \max_{i=1,2,\cdots,N_{test}} \left| y(x_i) - \widetilde{y}(x_i) \right|, \tag{56}$$

where N_{test} is the number of test points and x_i means the i-th input test point. $y(x)$ is the exact solution of the ruin probability equation and $\widetilde{y}(x)$ is the obtained approximate solution.

5.1. Numerical Example 1

In this experiment, we supposed that the probability of bankruptcy obeys the classical risk model and the claim amount follows an exponential distribution with mean $\mu = \frac{1}{\alpha}$, that is, for $x > 0$,

$$F(x) = 1 - e^{-\alpha x}. \tag{57}$$

In this case, the analytical solution to Equation (17) is known and for $u \geq 0$ can be expressed as:

$$\psi(u) = \frac{1}{1+\theta} e^{-\frac{\theta \alpha u}{1+\theta}}. \tag{58}$$

Similar to Lu et al. [57] and Zhou et al. [56], 21 training points are isometrically selected in the definition domain $[0, 10]$. Parameters in the ruin probability Equation (19) are set as $c = 3, \lambda = 4, \alpha = 2, b = 10$. Twelve basis functions are used and 20 equidistant points in $[0.25, 9.75]$ are chosen as test points set.

Figure 3 shows the exact ruin probability and approximate solution obtained by the BTENN with $N = 12$. It can be seen that the approximate solution and the exact solution are in good agreement. The absolute error obtained by the BTENN and the Legendre neural network (LNN) models are plotted in Figure 4. Compared with the LNN method, the numerical solution obtained by the BTENN model has smaller error fluctuations. Table 1 lists the numerical solutions of the BTENN at the test points and compares the errors of the approximate solutions obtained by different methods. The MSE of the BTENN is 2.2044×10^{-16}, while MAE is only 2.3783×10^{-08}, which is much smaller than the others. The accuracy of the BTENN approximate solution has been significantly improved, proving the effectiveness of the proposed method.

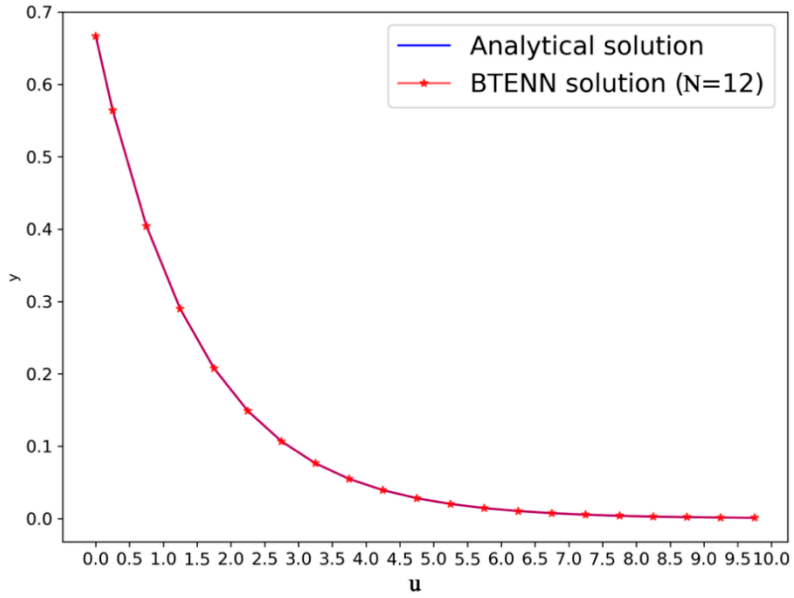

Figure 3. Analytical and numerical solutions obtained by the block trigonometric exponential neural network (BTENN).

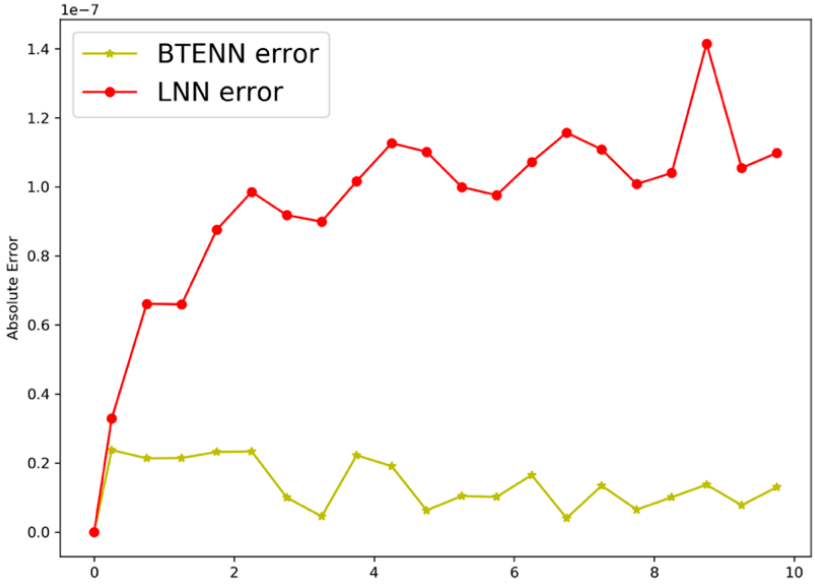

Figure 4. Comparison of absolute errors of approximate solution by BTENN and legendre neural network (LNN).

Moreover, according to Table 2, when 30 points are used for training, the MAE is 1.0836×10^{-08}. Further increasing the number of training points to 100 equidistant points, the MAE reduces to 8.1964×10^{-09}. It can be seen that the accuracy of the numerical solution increases as more training points are selected, but this improvement is not significant. Table 3 shows the mean square error and the maximum absolute error of the approximate solution obtained with different numbers of hidden layer neurons. When more hidden layer neurons are involved, a better approximation effect can be obtained, which is in line with the general approximation theorem in Remark 1.

Table 1. Comparison of Absolute Errors of Approximate Solutions Obtained by BTENN, LNN and Trigonometric Neural Networks (TNN).

u	Exact	Approximate	Absolute Error by BTENN	Absolute Error by LNN in [57]	Absolute Error by TNN in [56]
0.00	0.6666666667	0.6666666667	$0.0000 \times 10^{+0}$	$0.0000 \times 10^{+0}$	$0.0000 \times 10^{+0}$
0.25	0.5643211499	0.5643211737	2.3783×10^{-8}	3.2946×10^{-8}	4.2156×10^{-5}
0.75	0.4043537731	0.4043537518	2.1386×10^{-8}	6.6140×10^{-8}	7.9549×10^{-5}
1.25	0.2897321390	0.2897321605	2.1480×10^{-8}	6.5976×10^{-8}	9.5476×10^{-5}
1.75	0.2076021493	0.2076021261	2.3219×10^{-8}	8.7577×10^{-8}	1.0853×10^{-4}
2.25	0.1487534401	0.1487534167	2.3394×10^{-8}	9.8572×10^{-8}	1.1750×10^{-4}
2.75	0.1065864974	0.1065865074	1.0056×10^{-8}	9.1825×10^{-8}	1.2404×10^{-4}
3.25	0.0763725627	0.0763725672	4.5537×10^{-9}	8.9967×10^{-8}	1.2869×10^{-4}
3.75	0.0547233324	0.0547233102	2.2264×10^{-8}	1.0164×10^{-7}	1.3204×10^{-4}
4.25	0.0392109811	0.0392109620	1.9102×10^{-8}	1.1269×10^{-7}	1.3442×10^{-4}
4.75	0.0280958957	0.0280959020	6.3156×10^{-9}	1.1018×10^{-7}	1.3614×10^{-4}
5.25	0.0201315889	0.0201315994	1.0449×10^{-8}	9.9976×10^{-8}	1.3737×10^{-4}
5.75	0.0144249138	0.0144249036	1.0197×10^{-8}	9.7611×10^{-8}	1.3825×10^{-4}
6.25	0.0103359024	0.0103358859	1.6527×10^{-8}	1.0722×10^{-7}	1.3888×10^{-4}
6.75	0.0074059977	0.0074060017	4.0477×10^{-9}	1.1571×10^{-7}	1.3933×10^{-4}
7.25	0.0053066292	0.0053066427	1.3449×10^{-8}	1.1086×10^{-7}	1.3966×10^{-4}
7.75	0.0038023660	0.0038023595	6.5289×10^{-9}	1.0081×10^{-7}	1.3989×10^{-4}
8.25	0.0027245143	0.0027245042	1.0087×10^{-8}	1.0404×10^{-7}	1.4007×10^{-4}
8.75	0.0019521998	0.0019522135	1.3742×10^{-8}	1.4151×10^{-7}	1.4012×10^{-4}
9.25	0.0013988123	0.0013988045	7.7700×10^{-9}	1.0550×10^{-7}	1.4059×10^{-4}
9.75	0.0010022928	0.0010023058	1.3006×10^{-8}	1.0982×10^{-7}	1.3753×10^{-4}
MAE			2.3783×10^{-8}	1.4151×10^{-7}	1.4059×10^{-4}
MSE			2.2044×10^{-16}	1.0124×10^{-14}	1.6124×10^{-8}

Table 2. Error Comparison of Approximate Solutions with Different Numbers of Training Points.

Training Points	MSE	MAE
21	2.2044×10^{-16}	2.3783×10^{-8}
30	9.8427×10^{-17}	1.0836×10^{-8}
50	8.2830×10^{-17}	9.2712×10^{-9}
100	4.6419×10^{-17}	8.1964×10^{-9}

Table 3. Error Comparison of Approximate Solutions Using Different Numbers of Hidden Neurons.

Hidden Neurons	MSE	MAE
12	2.2044×10^{-16}	2.3783×10^{-8}
18	8.0732×10^{-17}	6.1296×10^{-9}
20	1.0714×10^{-18}	8.3127×10^{-10}
50	9.1445×10^{-22}	7.1498×10^{-12}

5.2. Numerical Example 2

This experiment illustrates the case when the claim amount follows an exponential distribution with mean $\mu = \frac{1}{\alpha}$ under the Erlang(2) risk model. Same as the previous experiment, we assume exponential claims with:

$$F(x) = 1 - e^{-\alpha x}, \tag{59}$$

for x > 0. An analytical solution exists for the ruin probability in this case and it can be expressed as:

$$\psi(u) = \frac{2\eta^2}{c^2\alpha^2 + 2\eta c\alpha + c\alpha\sqrt{(2\eta - c\alpha)^2 + 8\eta c\alpha - 4\eta^2}} e^{-\frac{2\eta^2 - c\alpha - \sqrt{(2\eta - c\alpha)^2 + 8\eta c\alpha - 4\eta^2}}{2c}u}, \tag{60}$$

Figure 5 describes the exact and the approximate solutions obtained by the proposed block trigonometric exponential neural network. Eight hidden neurons were used and contains the same parameters as Zhou [56], where $c = 3$, $\alpha = 1$, $\eta = 5$, $b = 10$ and 21 equidistant training points are chosen in the interval [0,10]. Table 4 presents the exact solution to the Equation (39) for the Erlang(2) model and the numerical solution obtained by the BTENN method at these test points. The absolute errors of the approximate solutions obtained by the BTENN, the LNN and the TNN are also compared in Table 4. As you can see, the MSE is 8.7029×10^{-18}, which is about ten-billionth of the error of the TNN model $O\left(10^{-08}\right)$. The absolute error curve of test points is plotted in Figure 6. The approximate solution error obtained by the proposed BTENN is lower, and the error of each test point is more stable. The absolute error does not become significantly larger as u increases. This shows that the proposed algorithm has good calculation stability. Moreover, it is obvious that among the selected test points, the minimum absolute error obtained by the LNN method is $O\left(10^{-08}\right)$, which is even greater than the maximum absolute error of the proposed BTENN method $O\left(10^{-09}\right)$.

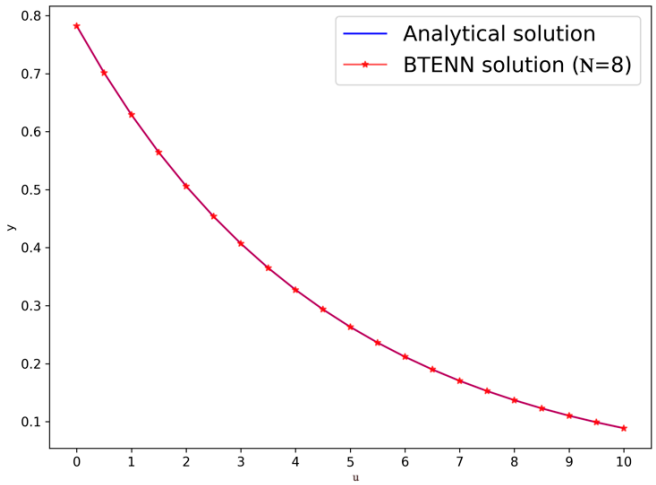

Figure 5. Analytical and Numerical Solutions Obtained by the BTENN of Example 2.

In addition, Figure 7 shows that even if the solution interval is extended to [0, 100], the numerical solution obtained by the block trigonometric exponential neural network containing only with eight hidden layer neurons is still close to the real solution. Based on the above facts, the BTENN method proposed in this study has the ability to accurately estimate the numerical solution of the ruin probability equation using only a small number of training points in the interval. Numerical experiments show that when insurance companies own more assets, the ruin probability decreases. However, too many assets will no longer significantly reduce its risk of ruin. Estimating the possibility of bankruptcy, allowing operators to use fewer assets and get a controllable risk.

Table 4. Comparison of the Numerical Solutions Obtained by BTENN, LNN and TNN.

u.	Exact	BTENN Solution	Absolute Error by BTENN	Absolute Error by LNN in [57]	Absolute Error by TNN in [56]
0.0	0.7822293562	0.7822293562	$0.0000 \times 10^{+0}$	$0.0000 \times 10^{+0}$	$0.0000 \times 10^{+0}$
0.5	0.7015293026	0.7015292999	2.7167×10^{-9}	2.4647×10^{-7}	2.8123×10^{-4}
1.0	0.6291548105	0.6291548124	2.0193×10^{-9}	4.5465×10^{-7}	2.8636×10^{-4}
1.5	0.5642469590	0.5642469609	1.8784×10^{-9}	1.0367×10^{-7}	1.5895×10^{-4}
2.0	0.5060354389	0.5060354357	3.3062×10^{-9}	3.4282×10^{-7}	2.6196×10^{-6}
2.5	0.4538294117	0.4538294097	2.1078×10^{-9}	2.8821×10^{-7}	1.3678×10^{-4}
3.0	0.4070093102	0.4070093132	3.0732×10^{-9}	8.5184×10^{-8}	2.1145×10^{-4}
3.5	0.3650194860	0.3650194894	3.4673×10^{-9}	3.3444×10^{-7}	2.1746×10^{-4}
4.0	0.3273616151	0.3273616137	1.4722×10^{-9}	2.3922×10^{-7}	1.6260×10^{-4}
4.5	0.2935887841	0.2935887798	4.3992×10^{-9}	7.1659×10^{-8}	6.6239×10^{-5}
5.0	0.2633001860	0.2633001850	1.0998×10^{-9}	2.9814×10^{-7}	4.5630×10^{-5}
5.5	0.2361363638	0.2361363676	3.7978×10^{-9}	2.4169×10^{-7}	1.4506×10^{-4}
6.0	0.2117749447	0.2117749479	3.3257×10^{-9}	3.5002×10^{-8}	2.0657×10^{-4}
6.5	0.1899268137	0.1899268117	1.9615×10^{-9}	2.6648×10^{-7}	2.1149×10^{-4}
7.0	0.1703326833	0.1703326792	4.2003×10^{-9}	2.1959×10^{-7}	1.5223×10^{-4}
7.5	0.1527600154	0.1527600159	3.2072×10^{-10}	6.6730×10^{-8}	3.6624×10^{-5}
8.0	0.1370002624	0.1370002664	4.0026×10^{-9}	2.6165×10^{-7}	1.0754×10^{-4}
8.5	0.1228663918	0.1228663912	3.5653×10^{-10}	7.4952×10^{-8}	2.2779×10^{-4}
9.0	0.1101906665	0.1101906634	3.1964×10^{-9}.	2.3406×10^{-7}	2.4184×10^{-4}
9.5	0.0988226546	0.0988226577	2.6166×10^{-9}	4.8428×10^{-8}	3.2086×10^{-5}
10.0	0.0886274433	0.0886274378	9.7183×10^{-10}	3.7546×10^{-7}	5.6048×10^{-4}
MAE			4.3992×10^{-9}	4.5465×10^{-7}	5.6048×10^{-4}
MSE			8.7029×10^{-18}	6.0264×10^{-7}	4.4036×10^{-8}

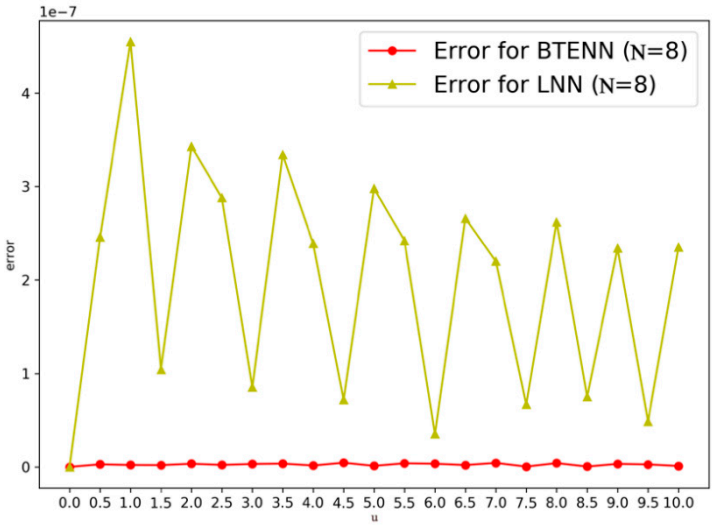

Figure 6. Absolute Error of the Numerical Solution of Example 2.

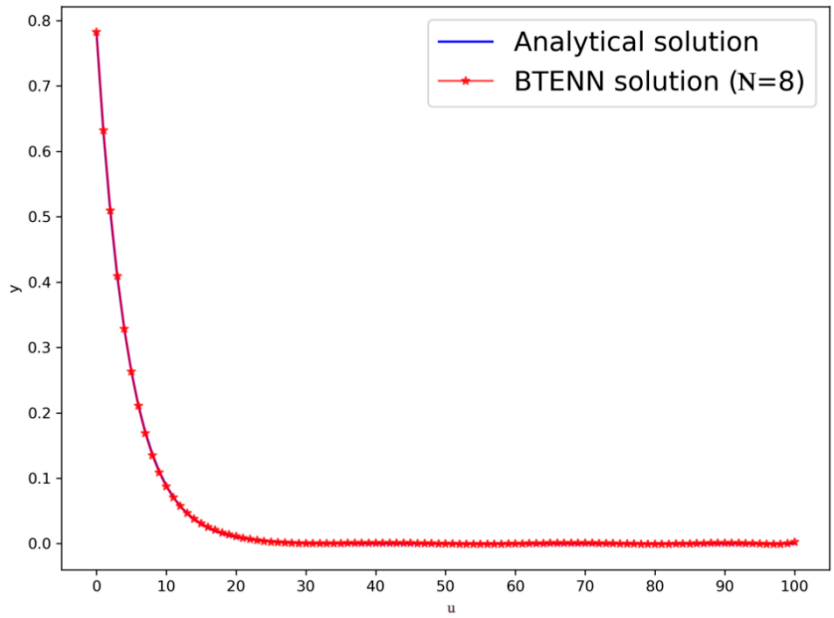

Figure 7. Analytical and Numerical Solutions Obtained on the Interval $[0,\ 100]$.

5.3. Numerical Example 3

The assumption that claim amounts follow a Pareto distribution, some studies was developed in some classical risk models for heavy-tail risks, where $X \sim Pareto(\alpha,\ \gamma)$ and $F(x)$ can be written as:

$$F(x) = 1 - \left(\frac{\gamma}{\gamma + x}\right)^{\alpha},\ F'(x) = \frac{\alpha}{\gamma}\left(\frac{\gamma}{\gamma + x}\right)^{\alpha+1},\ x > 0. \tag{61}$$

No analytical solution to Equation (19) exists in this case. In order to compare the approximate solutions obtained by the algorithm, we choose the same parameters $c = 600,\ \alpha = 3,\ \lambda = 1,\ \theta = 0.2,\ b = 10000$ as in Lu [57], with 51 equidistant points used as a training set for the BTENN model. Figure 8 plots the numerical solutions obtained by BTENN and LNN in the interval 0–10,000. The 50 test points depicted show that the approximate solutions obtained by the two algorithms almost overlap, which shows that the proposed BTENN model can be accurately calculated the ruin probability. It can be seen from the curve trend in Figure 8 that the ruin probability decreases as u rises, which is consistent with the actual situation. Table 5 contains solutions of these test points obtained by block trigonometric exponential neural network and trigonometric neural network. When the initial asset u is larger, the absolute difference between the two algorithms becomes larger. Considering that in numerical experiments 1 and 2, the error of LNN approximate solution increases when u is larger, which implies that the numerical solution obtained by the BTENN model is closer to the exact value of the ruin probability. The numerical solutions generated from the two algorithms are similar. The mean square difference is 3.2015×10^{-10}, which means that the proposed BTENN model can estimate the probability of ruin at any time, regardless of whether the ruin probability equation has an exact solution.

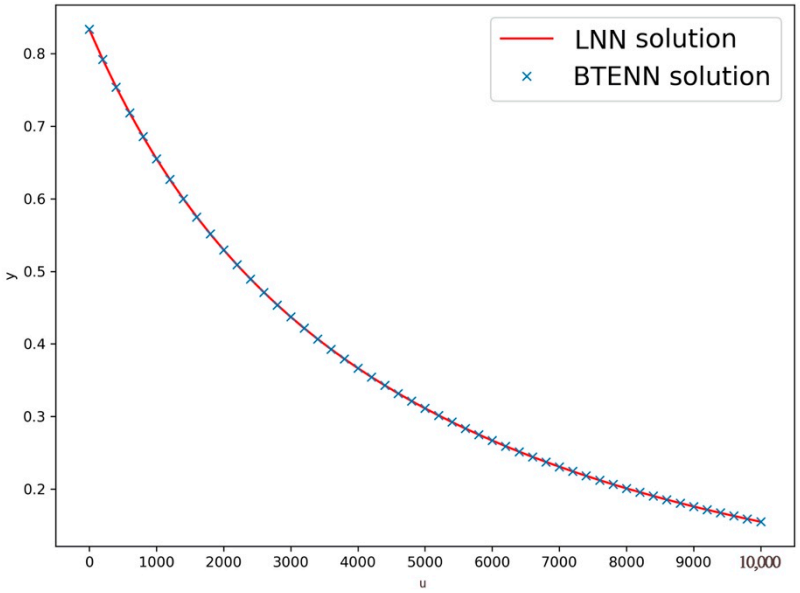

Figure 8. Numerical Solutions Obtained by the BTENN of Example 3.

Table 5. Comparison of the Numerical Solutions of the BTENN and TNN of Example 3.

u	BTENN Solution	LNN Solution [57]	Absolute Difference
230	0.7771674726	0.7771671970	2.7562×10^{-7}
1162	0.6262266038	0.6262251570	1.4468×10^{-6}
2094	0.5177610722	0.5177563420	4.7302×10^{-6}
3026	0.4367401507	0.4367394040	7.4668×10^{-7}
3958	0.3719139416	0.3719008950	1.3047×10^{-5}
4890	0.3188717066	0.3188620670	9.6396×10^{-6}
5822	0.2748805781	0.2748686660	1.1912×10^{-5}
6754	0.2380806874	0.2380170420	6.3645×10^{-5}
7686	0.2069896444	0.2069120930	7.7551×10^{-5}
8618	0.1805589985	0.1805008800	5.8118×10^{-5}
9550	0.1580081081	0.1579619740	4.6134×10^{-5}

6. Conclusion and Prospects

In this study, a block trigonometric exponential neural network is established to numerically solving the ruin probability equations. Three numerical experiments show that compared to the existing methods, the mean square error of the numerical solution obtained by the BTENN model is smaller and has a smaller maximum error. This shows that our BTENN model can accurately and in a stable way to calculate the ruin probability using only a few training points. Experiments also show that increasing the number of hidden neurons gives, approximate solutions with higher accuracy, which is consistent with the convergence theorem. In addition, the BTENN model can also be used in future to solve other types of continued integral equations. Future research can focus on simplified model calculations and promote the application range of the BTENN model.

Author Contributions: All authors contributed to the study conception and design. Y.C. (Yinghao Chen): Conceptualization, methodology, formal analysis, writing—original; C.Y.: conceptualization, methodology, visualization; X.X.: writing—review, project administration, funding acquisition; M.H.: conceptualization, writing—review & editing, supervision, funding acquisition; Y.C. (Yangjin Cheng): conceptualization. writing—review & editing. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Projects of National Social Science Foundation of China under Grant Np.19BT011.

Conflicts of Interest: All authors declared that they have no conflict of interests.

References

1. Drugdova, B. The issue of the commercial insurance, commercial insurance market and insurance of non-life risks. In *Financial Management of Firms and Financial Institutions: 10th International Scientific Conference, Pts I-Iv*; Culik, M., Ed.; Vsb-Tech Univ. Ostrava: Feecs, Czech Republic, 2015; pp. 202–208.
2. Opeshko, N.S.; Ivashura, K.A. Improvement of stress testing of insurance companies in view of european requirements. *Financ. Credit Act. Probl. Theory Pract.* **2017**, *1*, 112–119. [CrossRef]
3. Jia, F. *Analysis of State-owned holding Insurance Companies' Risk Management on the Basis of Equity Structure, 2nd China International Conference on Insurance and Risk Management (CICIRM)*; Tsinghua University Press: Beijing, China, 2011; pp. 60–63.
4. Cejkova, V.; Fabus, M. *Management and Criteria for Selecting Commercial Insurance Company for Small and Medium-Sized Enterprises*; Masarykova Univerzita: Brno, Czech Republic, 2014; pp. 105–110.
5. Belkina, T.A.; Konyukhova, N.B.; Slavko, B.V. Solvency of an Insurance Company in a Dual Risk Model with Investment: Analysis and Numerical Study of Singular Boundary Value Problems. *Comput. Math. Math. Phys.* **2019**, *59*, 1904–1927. [CrossRef]
6. Jin, B.; Yan, Q. *Diversification, Performance and Risk Taking of Insurance Company*; Tsinghua University Press: Beijing, China, 2013; pp. 178–188.
7. Wang, Y.; Yu, W.; Huang, Y.; Yu, X.; Fan, H. Estimating the Expected Discounted Penalty Function in a Compound Poisson Insurance Risk Model with Mixed Premium Income. *Mathematics* **2019**, *7*, 305. [CrossRef]
8. Song, Y.; Li, X.Y.; Li, Y.; Hong, X. Risk investment decisions within the deterministic equivalent income model. *Kybernetes* **2020**. [CrossRef]
9. Stellian, R.; Danna-Buitrago, J.P. Financial distress, free cash flow, and interfirm payment network: Evidence from an agent-based model. *Int. J. Financ. Econ.* **2019**. [CrossRef]
10. Emms, P.; Haberman, S. Asymptotic and numerical analysis of the optimal investment strategy for an insurer. *Insur. Math. Econ.* **2007**, *40*, 113–134. [CrossRef]
11. Zhu, S. A Becker-Tomes model with investment risk. *Econ. Theory* **2019**, *67*, 951–981. [CrossRef]
12. Jiang, W. Two classes of risk model with diffusion and multiple thresholds: The discounted dividends. *Hacet. J. Math. Stat.* **2019**, *48*, 200–212. [CrossRef]
13. Xie, J.-h.; Zou, W. On the expected discounted penalty function for the compound Poisson risk model with delayed claims. *J. Comput. Appl. Math.* **2011**, *235*, 2392–2404. [CrossRef]
14. Lundberg, F. *Approximerad Framställning Afsannollikhetsfunktionen: II. återförsäkring af Kollektivrisker*; Almqvist & Wiksells Boktr: Uppsala, Sweeden, 1903.
15. Andersen, E.S. On the collective theory of risk in case of contagion between claims. *Bull. Inst. Math. Appl.* **1957**, *12*, 275–279.
16. Dickson, D.C.M.; Hipp, C. On the time to ruin for Erlang(2) risk processes. *Insur. Math. Econ.* **2001**, *29*, 333–344. [CrossRef]
17. Li, S.M.; Garrido, J. On a class of renewal risk models with a constant dividend barrier. *Insur. Math. Econ.* **2004**, *35*, 691–701. [CrossRef]
18. Li, S.M.; Garrido, J. On ruin for the Erlang(n) risk process. *Insur. Math. Econ.* **2004**, *34*, 391–408. [CrossRef]
19. Gerber, H.U.; Yang, H. Absolute Ruin Probabilities in a Jump Diffusion Risk Model with Investment. *N. Am. Actuar. J.* **2007**, *11*, 159–169. [CrossRef]
20. Yazici, M.A.; Akar, N. The finite/infinite horizon ruin problem with multi-threshold premiums: A Markov fluid queue approach. *Ann. Oper. Res.* **2017**, *252*, 85–99. [CrossRef]
21. Lu, Y.; Li, S. The Markovian regime-switching risk model with a threshold dividend strategy. *Insur. Math. Econ.* **2009**, *44*, 296–303. [CrossRef]
22. Zhu, J.; Yang, H. Ruin theory for a Markov regime-switching model under a threshold dividend strategy. *Insur. Math. Econ.* **2008**, *42*, 311–318. [CrossRef]
23. Asmussen, S.; Albrecher, H. *Ruin Probabilities. Advanced Series on Statistical Science & Applied Probability*, 2nd ed.; World Scientific: Singapore, 2010; Volume 14, p. 630.
24. Wang, G.J.; Wu, R. Some distributions for classical risk process that is perturbed by diffusion. *Insur. Math. Econ.* **2000**, *26*, 15–24. [CrossRef]
25. Cai, J.; Yang, H.L. Ruin in the perturbed compound Poisson risk process under interest force. *Adv. Appl. Probab.* **2005**, *37*, 819–835. [CrossRef]

26. Bergel, A.I.; Egidio dos Reis, A.D. Ruin problems in the generalized Erlang(n) risk model. *Eur. Actuar. J.* **2016**, *6*, 257–275. [CrossRef]
27. Kasumo, C. Minimizing an Insurer's Ultimate Ruin Probability by Reinsurance and Investments. *Math. Comput. Appl.* **2019**, *24*, 21. [CrossRef]
28. Xu, L.; Wang, M.; Zhang, B. Minimizing Lundberg inequality for ruin probability under correlated risk model by investment and reinsurance. *J. Inequalities Appl.* **2018**. [CrossRef] [PubMed]
29. Zou, W.; Xie, J.H. On the probability of ruin in a continuous risk model with delayed claims. *J. Korean Math. Soc.* **2013**, *50*, 111–125. [CrossRef]
30. Andrulytė, I.M.; Bernackaitė, E.; Kievinaitė, D.; Šiaulys, J. A Lundberg-type inequality for an inhomogeneous renewal risk model. *Mod. Stoch. Theory Appl.* **2015**, *2*, 173–184. [CrossRef]
31. Fei, W.; Hu, L.; Mao, X.; Xia, D. Advances in the truncated euler-maruyama method for stochastic differential delay equations. *Commun. Pure Appl. Anal.* **2020**, *19*, 2081–2100. [CrossRef]
32. Li, F.; Cao, Y. Stochastic Differential Equations Numerical Simulation Algorithm for Financial Problems Based on Euler Method. In *2010 International Forum on Information Technology and Applications*; IEEE Computer Society: Los Alamitos, CA, USA, 2010; pp. 190–193. [CrossRef]
33. Zhang, C.; Qin, T. The mixed Runge-Kutta methods for a class of nonlinear functional-integro-differential equations. *Appl. Math. Comput.* **2014**, *237*, 396–404. [CrossRef]
34. Cardoso, R.M.R.; Waters, H.R. Calculation of finite time ruin probabilities for some risk models. *Insur. Math. Econ.* **2005**, *37*, 197–215. [CrossRef]
35. Makroglou, A. Computer treatment of the integro-differential equations of collective non-ruin; the finite time case. *Math. Comput. Simul.* **2000**, *54*, 99–112. [CrossRef]
36. Paulsen, J.; Kasozi, J.; Steigen, A. A numerical method to find the probability of ultimate ruin in the classical risk model with stochastic return on investments. *Insur. Math. Econ.* **2005**, *36*, 399–420. [CrossRef]
37. Tsitsiashvili, G.S. Computing ruin probability in the classical risk model. *Autom. Remote Control* **2009**, *70*, 2109–2115. [CrossRef]
38. Zhang, Z. Approximating the density of the time to ruin via fourier-cosine series expansion. *Astin Bull.* **2017**, *47*, 169–198. [CrossRef]
39. Muzhou Hou, Y.C. Industrial Part Image Segmentation Method Based on Improved Level Set Model. *J. Xuzhou Inst. Technol.* **2019**, *40*, 10.
40. Wang, Z.; Meng, Y.; Weng, F.; Chen, Y.; Lu, F.; Liu, X.; Hou, M.; Zhang, J. An Effective CNN Method for Fully Automated Segmenting Subcutaneous and Visceral Adipose Tissue on CT Scans. *Ann. Biomed. Eng.* **2020**, *48*, 312–328. [CrossRef] [PubMed]
41. Hou, M.; Zhang, T.; Weng, F.; Ali, M.; Al-Ansari, N.; Yaseen, Z.M. Global solar radiation prediction using hybrid online sequential extreme learning machine model. *Energies* **2018**, *11*, 3415. [CrossRef]
42. Muzhou Hou, T.Z.; Yang, Y.; Luo, J. Application of Mec- based ELM algorithm in prediction of PM2.5 in Changsha City. *J. Xuzhou Inst. Technol.* **2019**, *34*, 1–6.
43. Hahnel, P.; Marecek, J.; Monteil, J.; O'Donncha, F. Using deep learning to extend the range of air pollution monitoring and forecasting. *J. Comput. Phys.* **2020**, *408*. [CrossRef]
44. Chen, Y.; Xie, X.; Zhang, T.; Bai, J.; Hou, M. A deep residual compensation extreme learning machine and applications. *J. Forecast.* **2020**, 1–14. [CrossRef]
45. Weng, F.; Chen, Y.; Wang, Z.; Hou, M.; Luo, J.; Tian, Z. Gold price forecasting research based on an improved online extreme learning machine algorithm. *J. Ambient Intell. Humaniz. Comput.* **2020**. [CrossRef]
46. Hou, M.; Liu, T.; Yang, Y.; Zhu, H.; Liu, H.; Yuan, X.; Liu, X. A new hybrid constructive neural network method for impacting and its application on tungsten price prediction. *Appl. Intell.* **2017**, *47*, 28–43. [CrossRef]
47. Hou, M.; Han, X. Constructive Approximation to Multivariate Function by Decay RBF Neural Network. *IEEE Trans. Neural Netw.* **2010**, *21*, 1517–1523. [CrossRef]
48. Sun, H.; Hou, M.; Yang, Y.; Zhang, T.; Weng, F.; Han, F. Solving Partial Differential Equation Based on Bernstein Neural Network and Extreme Learning Machine Algorithm. *Neural Process. Lett.* **2019**, *50*, 1153–1172. [CrossRef]
49. Yang, Y.; Hou, M.; Luo, J. A novel improved extreme learning machine algorithm in solving ordinary differential equations by Legendre neural network methods. *Adv. Differ. Equ.* **2018**, *2018*, 469. [CrossRef]
50. Yang, Y.; Hou, M.; Luo, J.; Liu, T. Neural Network method for lossless two-conductor transmission line equations based on the IELM algorithm. *AIP Adv.* **2018**, *8*. [CrossRef]

51. Yang, Y.; Hou, M.; Sun, H.; Zhang, T.; Weng, F.; Luo, J. Neural network algorithm based on Legendre improved extreme learning machine for solving elliptic partial differential equations. *Soft Comput.* **2020**, *24*, 1083–1096. [CrossRef]
52. Sabir, Z.; Wahab, H.A.; Umar, M.; Sakar, M.G.; Raja, M.A.Z. Novel design of Morlet wavelet neural network for solving second order Lane-Emden equation. *Math. Comput. Simul.* **2020**, *172*, 1–14. [CrossRef]
53. Hure, C.; Pham, H.; Warin, X. Deep backward schemes for high-dimensional nonlinear pdes. *Math. Comput.* **2020**, *89*, 1547–1579. [CrossRef]
54. Samaniego, E.; Anitescu, C.; Goswami, S.; Nguyen-Thanh, V.M.; Guo, H.; Hamdia, K.; Zhuang, X.; Rabczuk, T. An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. *Comput. Methods Appl. Mech. Eng.* **2020**, *362*. [CrossRef]
55. Huang, G.; Huang, G.B.; Song, S.; You, K. Trends in extreme learning machines: A review. *Neural Netw.* **2015**, *61*, 32–48. [CrossRef]
56. Zhou, T.; Liu, X.; Hou, M.; Liu, C. Numerical solution for ruin probability of continuous time model based on neural network algorithm. *Neurocomputing* **2019**, *331*, 67–76. [CrossRef]
57. Lu, Y.; Chen, G.; Yin, Q.; Sun, H.; Hou, M. Solving the ruin probabilities of some risk models with Legendre neural network algorithm. *Digit. Signal Process.* **2020**, *99*. [CrossRef]
58. Zhang, X.; Wu, J.; Yu, D. Superconvergence of the composite Simpson's rule for a certain finite-part integral and its applications. *J. Comput. Appl. Math.* **2009**, *223*, 598–613. [CrossRef]

Availability of Data and Material: Not applicable.
Code Availability: All code is executed by Python3.6, mail to chenyinghao@csu.edu.cn.