Solving Partial Differential Equations with Bernstein Neural Networks

Sina Razvarz [1], Raheleh Jafari [2], Alexander Gegov[3]

[1] Departamento de Control Automatico CINVESTAV-IPN (National Polytechnic Institute)
Mexico City, Mexico
srazvarz@yahoo.com
[2] Center for Artificial Intelligence Research (CAIR), University of Agder, Grimstad, Norway
Jafari3339@yahoo.com
[3] School of Computing, University of Portsmouth, Buckingham Building, Portsmouth PO13HE, UK
alexander.gegov@port.ac.uk

Abstract. In this paper, a neural network-based procedure is suggested to produce estimated solutions (controllers) for the second-order nonlinear partial differential equations (PDEs). This concept is laid down so as to produce a prevalent approximation on the basis of the learning method which is at par with quasi-Newton rule. The proposed neural network contains the regularizing parameters (weights and biases), that can be utilized for making the error function least. Besides, an advanced technique is presented for resolving PDEs based on the usage of Bernstein polynomial. Numerical experiments alongside comparisons show the fantastic capacity of the proposed techniques.

Keywords: Neural Network, Bernstein Polynomial, Partial Differential Equations.

1 Introduction

Exact solution of differential equation plays a noteworthy role in the fitting seizing of qualitative characters of several processes as well as occurrences at par with several zones of natural science. Exact solutions authorize researchers for designing and carrying out experiments by developing valid natural conditions for determining these parameters or functions. However, obtaining the exact solutions of the PDEs is complicated and is case specific.

Several techniques have been proposed in literature in order to resolve different types of PDEs. In [1] the Homotopy perturbation technique is utilized to obtain the solution of PDEs with variable coefficients. In [2] the resolving of the PDEs requires two-dimensional differential transformation techniques. In [3] the modified technique of simplest equation is employed for extracting exact analytical solutions of nonlinear PDEs. In [4] an iteration technique in order to solve both linear as well as nonlinear wave equations is analyzed. In the following, some numerical solutions are laid down that have been suggested by other researchers. In [5], implementation of a spreadsheet program produces the numerical solution of the hyperbolic equation. Some researchers also generate an array solution that includes the value of the solution at a selected group of points [6].

Evje et al. [7] utilized an explicit monotone difference technique in order to estimate the entropy solutions related to degenerate parabolic equation. Bulbul et al. [8] proposed a Taylor polynomial estimation for the solution of hyperbolic type PDEs having constant coefficients. In [9], the researchers analyzed double non-traveling wave solutions of two systems associated with nonlinear PDEs. In [10] convolution quadrature is employed to the time-domain boundary integral formulation related to the wave equation having non-zero initial conditions. In [11], Martinez worked on a linear wave equation having a boundary damping term. Catania et al. For prevailing results related to the feedback control of the wave equation we suggest [12] and for open-loop control of the wave equation we refer [13]. However, the above discussed techniques are very complicated. Since the solutions associated with PDE are considered to be uniformly continuous, also the problems linked to the viable sets are usually compact, neural networks are best suited candidates in order to estimate viability problems [14].

Neural network finds its application in the fields of mathematics, chemistry, physics, and numerous applications [15-20]. They have become recently popular for solving PDEs. In [21] a feed-forward neural network is laid down in order to resolve an elliptic PDE in 2D. Another methodology is proposed in [22] for resolving a class of first-order PDEs on the basis of Multilayer neural networks. In [23] an unsupervised neural network is suggested in order to solve the nonlinear Schrodinger equation. In [24] the solutions of vibration control problems by utilizing artificial neural networks is discussed. In [25] a controlled heat problem up to three decimal digits accuracy is resolved by utilizing feed forward neural networks.

In this paper, a methodology based on neural networks is proposed in order to solve the strongly degenerate parabolic equations. The trial solution related to the PDE is stated as an addition of two parts. The primary part suffices the initial and boundary conditions, and does not have adjustable parameters. The secondary part includes a neural network having adjustable parameters (weights and biases). Furthermore, a superiorly modified technique is laid down in order to solve a wave equation in concerned with the application of Bernstein neural networks, that contains an excellent attributes of Bernstein polynomial. The Bernstein polynomial extracts its position in the theory of estimation considering the fact that it has good uniform approximation capability for continuous functions. These polynomials are suitable to produce a smooth estimation for equal distance knots [26]. The implementation of Bernstein polynomials is suggested in this paper, because it is extensively simple to apply. Also it is continuously differentiable due to the nature of theoretical contents.

The rest of this paper is organized as follows. Section 2 provides a summarized description of PDEs. An innovative method for resolving PDEs based on the neural networks is proposed in this section. Also, an advanced method is supplied for solving PDEs based on the application of a Bernstein neural network named as dynamic model. Experiments, simulation results, and comparisons are completed and discussed in Section 3. Section 4 concludes the paper.

2 Nonlinear system modeling with partial differential equations

Definition 1 (Second-order nonlinear PDE) The second-order singular nonlinear PDE can be illustrated by utilizing the equation mentioned below

$$\frac{\partial^2 v(x,t)}{\partial t^2}+\frac{2}{t}\frac{\partial v(x,t)}{\partial t}=F(x,v(x,t),\frac{\partial v(x,t)}{\partial x},\frac{\partial^2 v(x,t)}{\partial x^2}) \qquad (1)$$

in which t and x are independent variables, v is the dependent variable, F is a nonlinear function of x, v, v_x and v_{xx}, also the initial conditions for the PDE (1) are illustrated as below

$$v(x,0)=f(x), \quad v_t(x,0)=g(x)$$

Definition 2 (Strongly degenerate parabolic equations) The strongly degenerate parabolic equation is described as

$$v_t+Q(v)_x=A(v)_{xx}, (x,t)\in \Pi_T := [0,1]\times(0,T), \quad T>0 \qquad (2)$$

with boundary conditions

$$v(x,0)=g_0(x),\ v(0,t)=f_0(t),\ v(1,t)=f_1(t) \qquad (3)$$

in which the integrated diffusion coefficient A is demonstrated by

$$A(v)=\int_0^v a(s)ds,\ \ a(v)\geq 0, a\in L^\infty([0,1])\cap L^1([0,1]) \qquad (4)$$

The function a is permitted to disappear on v-intervals of positive length, on which Eq. (2) degenerates to a first-order scalar conservation law. Therefore, Eq. (2) is named as strongly degenerate.

Definition 3 (wave equation) The Cauchy problem related to the wave equation in one dimension is defined as

$$\frac{\partial^2 v(x,t)}{\partial t^2}=c^2\frac{\partial^2 v(x,t)}{\partial x^2},\ \ (x,t)\in[0,a]\times[0,b] \qquad (5)$$

with

$$v(x,0)=\phi(x), \quad v_t(x,0)=\psi(x)$$

where a as well as b are constants. In the above mentioned equation, the parameter c is termed as the speed of light.

2.1 Controller design with neural networks approximation

Here, we construct a neural network for resolving the strongly degenerate parabolic equation that obtains the solution of differential equations in a closed analytic and differentiable form (Figure 1). The relation between the input-output of each unit in the suggested neural architecture is described as follows

- Input units

$$o_1^1 = x,\ o_2^1 = t$$

- Hidden units

$$o_j^2 = F\left(b_j + w_j^1 x + w_j^2 t\right),\ \ j = 1, \dots, m$$

- Output unit:

$$N(x,t) = \sum_{j=1}^{m} (W_j o_j^2)$$

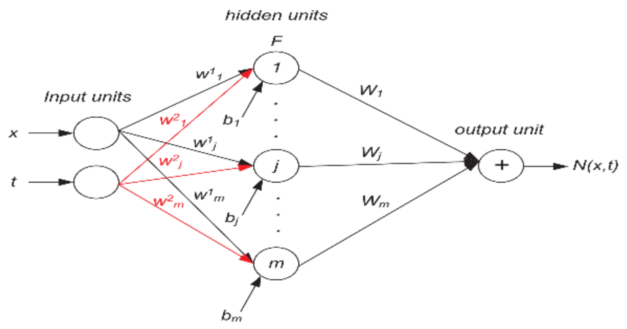

Fig. 1. Neural network equivalent to strongly degenerate parabolic equations

The activation function of the hidden units in this neural network is $F(r) = \frac{2}{1+e^{-2r}} - 1$ (tan-sigmoid function). The trial solution associated with (2) is portrayed as

$$v_m(x,t) = (1-x)f_0(t) + xf_1(t) + (1-t)\{g_0(x) - [(1-x)g_0(0) + xg_0(1)]\} + x(1-x)tN(x,t)$$

where

$$N(x,t) = \sum_{j=1}^{m} (W_j F(b_j + w_j^1 x + w_j^2 t))$$

The least mean square error is obtained for $(x,t) \ = \ (x_i\,, t_j)$ as below

$$E_{i,j} = \frac{1}{2}(M_{i,j})^2$$

where

$$M_{i,j} = \frac{\partial v_m(x,t)}{\partial t}\Big|_{\substack{x=x_i\\ t=t_j}} + \frac{\partial Q(v_m(x,t))}{\partial x}\Big|_{\substack{x=x_i\\ t=t_j}} - \frac{\partial^2 A(v_m(x,t))}{\partial x^2}\Big|_{\substack{x=x_i\\ t=t_j}}$$

In order to adjust the parameters we utilize Newton's rule. The standard self-learning process works as below

$$W_q(r+1) = W_q(r) - \mu(r)\frac{\partial E_{i,j}}{\partial w_q} + \gamma[W_q(r) - W_q(r-1)] \tag{6}$$

Now, the explicit technique of Eq. (6) is illustrated as

$$\begin{bmatrix} W_1 \\ \vdots \\ W_m \end{bmatrix}_{r+1} = \begin{bmatrix} W_1 \\ \vdots \\ W_m \end{bmatrix}_r - \frac{\left(\nabla E_{i,j}(W)_r\right)^T \nabla E_{i,j}(W)_r}{\left(\nabla E_{i,j}(W)_r\right)^T Q_r \nabla E_{i,j}(W)_r} \nabla E_{i,j}(W)_r + \gamma \begin{bmatrix} \Delta W_1 \\ \vdots \\ \Delta W_m \end{bmatrix}_{r-1} \tag{7}$$

where

$$\nabla E_{i,j}(W) = \left(\frac{\partial E_{i,j}}{\partial W_1}, \dots, \frac{\partial E_{i,j}}{\partial W_m}\right)^T$$

and

$$Q = \begin{bmatrix} \frac{\partial^2 E_{i,j}}{\partial W_1^2} & \frac{\partial^2 E_{i,j}}{\partial W_1 \partial W_2} & \cdots & \frac{\partial^2 E_{i,j}}{\partial W_1 \partial W_m} \\ \frac{\partial^2 E_{i,j}}{\partial W_2 \partial W_1} & \frac{\partial^2 E_{i,j}}{\partial W_2^2} & \cdots & \frac{\partial^2 E_{i,j}}{\partial W_2 \partial W_m} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 E_{i,j}}{\partial W_m \partial W_1} & \frac{\partial^2 E_{i,j}}{\partial W_m \partial W_2} & \cdots & \frac{\partial^2 E_{i,j}}{\partial W_m^2} \end{bmatrix}$$

The chain rule for differentiation can be illustrated as

$$\frac{\partial E_{i,j}}{\partial W_q} = \frac{\partial E_{i,j}}{\partial M_{i,j}} \cdot \frac{\partial M_{i,j}}{\partial W_q} = M_{i,j}\frac{\partial M_{i,j}}{\partial W_q}$$

The above learning procedure can be extended to other network parameters (w_q^1, w_q^2 and b_q) in a same way.

2.2 Controller design with Bernstein neural networks approximation

We carry forward our strategy for resolving wave equations by utilizing two patterns of Bernstein neural networks. Let us take into consideration the Cauchy problem (5), where the solution v relies on both spatial as well as temporal variables x and t respectively. The trial solution associated with (5) in the form of the Bernstein neural network is portrayed as

$$v_m(x,t) = \phi(x) + t\psi(x) + t\left[B(x,t) - B(x,0) - \frac{\partial B(x,0)}{\partial t}\right]$$

where $B(x,t)$ is the bivariate Bernstein polynomial series of solution function $v(x,t)$, termed as

$$B(x,t) = \sum_{i=0}^{n}\sum_{j=0}^{m} \binom{n}{i}\binom{m}{j} \frac{x^i(a-x)^{n-i}}{a^n}\frac{t^j(b-t)^{m-j}}{b^m} q_{i,j}(x,t), \quad n,m \in N$$

$q_{i,j}$ is the coefficient. We can state the above relation as

$$B(x,t) = \sum_{i=0}^{n}\sum_{j=0}^{m} \beta_{i,j}\, x^i(a-x)^{n-i} t^j(b-t)^{m-j} q_{i,j}(x,t), \quad n,m \in N \ , \beta_{i,j} = \frac{1}{a^n b^m}\binom{n}{i}\binom{m}{j} \tag{8}$$

where $\binom{n}{i} = \frac{n!}{i!(n-i)!}$ and $\binom{m}{j} = \frac{m!}{j!(m-j)!}$
Taking into account the follow relations

$$\frac{\partial^2 v_m(x,t)}{\partial x^2} = \phi''(x) + t\psi''(x) + t[\frac{\partial^2 B(x,t)}{\partial x^2} - \frac{\partial^2 B(x,0)}{\partial x^2} - \frac{\partial^2 \partial B(x,0)}{\partial x^2 \partial t}]$$

and

$$\frac{\partial^2 v_m(x,t)}{\partial t^2} = 2\frac{\partial B(x,t)}{\partial t} + t\frac{\partial^2 B(x,t)}{\partial t^2}$$

replacing the above relations in the origin problem (5) gives us the following differential equation

$$2\frac{\partial B(x,t)}{\partial t} + t\frac{\partial^2 B(x,t)}{\partial t^2} = c^2\left(\phi''(x) + t\psi''(x) + t\left[\frac{\partial^2 B(x,t)}{\partial x^2} - \frac{\partial^2 B(x,0)}{\partial x^2} - \frac{\partial^2 \partial B(x,0)}{\partial x^2 \partial t}\right]\right) \quad (9)$$

$$(x,t) \in [0,a] \times [0,b]$$

For simplicity the above relation can be justified as mentioned below

$$\sum_{i=0}^{n}\sum_{j=0}^{m} \xi_{i,j}(x,t)\, q_{i,j}(x,t) = \zeta(x,t), \quad (x,t) \in [0,a] \times [0,b] \quad (10)$$

where

$\xi_{i,j}(x,t) = 2\beta_{i,j}x^i(a-x)^{n-i}(jt^{j-1}(b-t)^{m-j} - (m-j)t^j(b-t)^{m-j-1}) + t\beta_{i,j}x^i(a-x)^{n-i}(j(j-1)t^{j-2}(b-t)^{m-j} - 2j(m-j)t^{j-1}(b-t)^{m-j-1} + (m-j)(m-j-1)t^j(b-t)^{m-j-2}) + c^2 t\beta_{i,j}(i(i-1)x^{i-2}(a-x)^{n-i} - 2i(n-i)x^{i-1}(a-x)^{n-i-1} + (n-i)(n-i-1)x^i(a-x)^{n-i-2})t^j(b-t)^{m-j} - c^2 t\beta_{i,j}(i(i-1)x^{i-2}(a-x)^{n-i} - 2i(n-i)x^{i-1}(a-x)^{n-i-1} + (n-i)(n-i-1)x^i(a-x)^{n-i-2})(jt^{j-1}(b-t)^{m-j} - (m-j)t^j(b-t)^{m-j-1})$

and

$$\zeta(x,t) = c^2\left(\phi''(x) + t\psi''(x) + t\frac{\partial^2 B(x,0)}{\partial x^2}\right)$$

The Bernstein neural network (8) is shown in Figure 2.
In the above architecture the mathematical symbol is defined as

$$\varphi_{i,j} = \sum_{i=0}^{n}\sum_{j=0}^{m}\binom{n}{i}\binom{m}{j}\frac{x^i(a-x)^{n-i}}{a^n}\frac{t^j(b-t)^{m-j}}{b^m}, \quad n,m \in N$$

The relation between the input-output of each unit in the suggested neural architecture is described as follows

- Input unit

$$o_{i,j} = q_{i,j} \quad i = 0,\dots,n, \ \ j = 0,\dots,m$$

- Output unit

$$N(x,t) = \varphi_{i,j} o_{i,j}$$

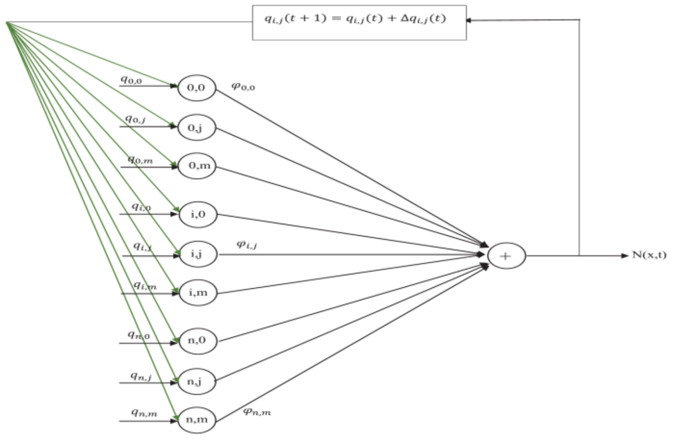

Fig. 2. Dynamic Bernstein neural network

Now, an appropriate numerical methodology should be capable enough to supply a suitable tool in order to measure and compute the preciseness of the extracted solution. Laying down the cost function $E_{i,j}$ over the model parameters makes it a good predictor. The least mean square error is stated to be as one of the most talked usable cost function associated with $E_{i,j}$. Assume that $0 \leq x \leq 1$, the rectangle $[0,1] \times [0,T]$is divided into nn' mesh points$(x_i, t_j) = ((i-1)h, (j-1)h'), h = \frac{1}{n-1}, h' = \frac{T}{n'-1}, (i = 1, \dots, n; j = 1, \dots, n')$.Therefore for comparing the exact solution with its extracted one, the least mean square error is utilized that is described as mentioned below

$$E_{i,j} = \frac{1}{2}(\sum_{i=0}^{n}\sum_{j=0}^{m} \xi_{i,j}(x,t) q_{i,j}(x,t) - \zeta(x,t))^2$$

We utilize Newton's rule as mentioned in (12) in order to adjust the parameters in such a manner that the network error attains minimal over the space of weights setting. The initial parameter $q_{i,j}$ is chosen on a random basis in order to start the procedure. The illustrated standard self-learning process works as below

$$q_{i,j}(r+1) = q_{i,j}(r) - \mu(r)\frac{\partial E_{i,j}}{\partial q_{i,j}}$$

where μ is the training rate $\mu > 0$. For increasing the training process, a momentum term is added up as follows

$$q_{i,j}(r+1) = q_{i,j}(r) - \mu(r)\frac{\partial E_{i,j}}{\partial q_{i,j}} + \gamma[q_{i,j}(r) - q_{i,j}(r-1)] \tag{11}$$

where $\gamma > 0$. The index r is the iteration number. Also, $q_{i,j}(r+1)$ and $q_{i,j}(r)$ represents the updated and recent output weight values, respectively. Now, the explicit technique of Eq. (11) is illustrated as

$$\begin{bmatrix} q_{0,0} \\ \vdots \\ q_{n,m} \end{bmatrix}_{r+1} = \begin{bmatrix} q_{0,0} \\ \vdots \\ q_{n,m} \end{bmatrix}_{r} - \frac{\left(\nabla E_{i,j}(q)_r\right)^T \nabla E_{i,j}(q)_r}{\left(\nabla E_{i,j}(q)_r\right)^T Q_r \nabla E_{i,j}(q)_r} \nabla E_{i,j}(q)_r + \gamma \begin{bmatrix} \Delta q_{0,0} \\ \vdots \\ \Delta q_{n,m} \end{bmatrix}_{r-1} \tag{12}$$

where

$$\nabla E_{i,j}(q) = \left(\frac{\partial E_{i,j}}{\partial q_{0,0}}, \ldots, \frac{\partial E_{i,j}}{\partial q_{n,m}}\right)^T$$

and

$$Q = \begin{bmatrix} \frac{\partial^2 E_{i,j}}{\partial q_{0,0}^2} & \frac{\partial^2 E_{i,j}}{\partial q_{0,0}\partial q_{1,1}} & \cdots & \frac{\partial^2 E_{i,j}}{\partial q_{0,0}\partial q_{n,m}} \\ \frac{\partial^2 E_{i,j}}{\partial q_{1,1}\partial q_{0,0}} & \frac{\partial^2 E_{i,j}}{\partial q_{1,1}^2} & \cdots & \frac{\partial^2 E_{i,j}}{\partial q_{1,1}\partial q_{n,m}} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 E_{i,j}}{\partial q_{n,m}\partial q_{0,0}} & \frac{\partial^2 E_{i,j}}{\partial q_{n,m}\partial q_{1,1}} & \cdots & \frac{\partial^2 E_{i,j}}{\partial q_{n,m}^2} \end{bmatrix} \tag{13}$$

are calculated at the current mesh points (x_i, t_j) In this case, Q is the Hessian matrix with components $\frac{\partial^2 E_{i,j}}{\partial q_{i,j}\partial q_{\tilde{i},\tilde{j}}}$ (for $i, \tilde{i} = 0, \ldots, n; j, \tilde{j} = 0, \ldots, m$). It is clear that the convergence speed is in direct relation at par with the learning rate parameter. For attaining the optimal learning rate in concerned with rapid convergence of our learning optimization rule, the inverse of Hessian matrix Q of the error function $E_{i,j}$is introduced at the current mesh points. The Newton technique mentioned above is very much capable for scaling the descent step in each step. Now, the partial derivative $\frac{\partial E_{i,j}}{\partial q_{i,j}}$ can be extracted at the current weight values.

3 Numerical results and discussion

All computations mentioned in the following tables are obtained by utilizing Matlab.

Example 1 The Buckley-Leverett differential equation is portrayed as [27]

$$\frac{\partial v(x,t)}{\partial t} + \frac{\partial g(v(x,t))}{\partial x} = \frac{\partial^2 A(v(x,t))}{\partial x^2}$$

Where

$$g(v) = \frac{v^2}{v^2 + (1-v)^2}, a(v) = 4\varepsilon v(1-v),$$

on the domain $(x,t) \in [0,1] \times [0,0.5]$ with initial condition

$$v_0(x) = \begin{cases} 0 & x < 0.1 \\ 1 & otherwise \end{cases}$$

and boundary conditions

$$v(0,t) = 1, \qquad v(1,t) = 0$$

also, $\varepsilon = 0.01$

The following characteristics are taken into consideration

I. $Time\ step: k' = 0.98\frac{k^2}{k\|g'\|_\infty + 2\varepsilon\|a\|_\infty}$

II. L^1-error $E_{mid} = \left(\sum_{i=1}^{n}\sum_{j=1}^{n'}\left|v(x_i,t_j)\right|\right)^{-1}\sum_{i=1}^{n}\sum_{j=1}^{n'}\left|v(x_i,t_j) - v_m(x_i,t_j)\right|$

where $v_m(x_i,t_j)$ as well as $v(x_i,t_j)$ are termed as the calculated solution and exact value of reference solution at grid point (x_i,t_j), respectively.

This problem is solved by applying the methodology of neural network proposed in this paper. Comparisons between the suggested algorithm on different training steps and the discrete mollification scheme [27] with support parameter ϑ are displayed in Table 1.

Table 1. Approximation errors of neural network based algorithm and mollified method

1/k	**Mollified method**			**r**	**NN method**		
	$\vartheta = 3$	$\vartheta = 5$	$\vartheta = 8$		$m = 7$	$m = 11$	$m = 17$
64	**2.6105e-2**	**2.5327e-2**	**2.5055e-2**	**15**	**2.4021e-2**	**1.8035e-2**	**1.0012e-2**
128	**1.4932e-2**	**1.4287e-2**	**1.4133e-2**	**30**	**8.0207e-3**	**6.8025e-3**	**5.5215e-3**
256	**8.3709e-3**	**7.9698e-3**	**7.6833e-3**	**45**	**1.9548e-3**	**1.0008e-3**	**8.9025e-4**
512	**4.5075e-3**	**4.3271e-3**	**4.1141e-3**	**60**	**4.9925e-4**	**3.1875e-4**	**1.9011e-4**
1024	**1.9997e-3**	**1.9335e-3**	**1.8279e-3**	**75**	**7.3081e-5**	**5.8295e-5**	**4.6952e-5**

Example 2 The following wave equation is taken into consideration that models the motion associated with the guitar string of length L

$$\frac{\partial^2 v(x,t)}{\partial t^2} = c^2\frac{\partial^2 v(x,t)}{\partial x^2}$$

with the boundary conditions

$$v(0,t) = sin(\pi t), \quad v(L,t) = 0$$

on the domain $(x,t) \in [0,L] \times [0,T]$ initial position and velocity

$$v(x,0) = 0, \;\; v_t(x,0) = \pi\cos(\pi x)$$

In the proposed problem $c^2 = \frac{T_s}{\rho}$,where T_s is taken to be the tension in the string, also ρ is the density of the string. The specifications mentioned here are given by $L = 1, T = 4, T_s = 2\frac{N}{m}$ and $\rho = 2\frac{kg}{m^3}$. The exact solution related to the problem is $v(x,t) = \cos(\pi x)\sin(\pi t)$.

We use dynamic Bernstein neural network (DNN) to approximate the solution. To compare our results, we use the other two popular methods: 3-point explicit method and optimal explicit method [28]. The exact solution is displayed in Figure 3. Corresponding approximated error plots are shown in Figure 4.

Example 3 Two semi-infinite strings of different densities are joined as [29]

$$\frac{\partial^2 v(x,t)}{\partial t^2} = (c_1^2 + c_2^2)\frac{\partial^2 v(x,t)}{\partial x^2}$$

with the boundary conditions

$$v(0,t) = \cos(\pi t), \;\; v(L,t) = 0$$

On the domain $(x,t) \in [0,T]$ initial position and velocity

$$v(x,0) = \cos(\pi x), \;\; v_t(x,0) = 0$$

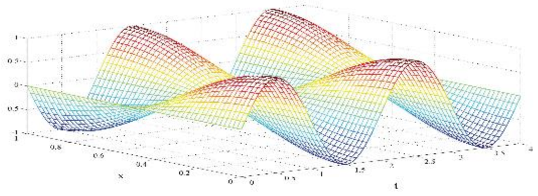

Fig. 3. Plot of the exact solution

see Figure 5. In the proposed problem $c = \sqrt{\frac{E}{\rho}}$, where E is the Young's modulus, also ρ is the density of the rod. The specifications mentioned here are given by $L = 1$, $T = 5, E_1 = 2\frac{kg}{m.s^2}$,$\rho_1 = 2.882\frac{kg}{m^3}$,$E_2 = 4.3\frac{kg}{m.s^2}$ and $\rho_1 = 15.136\frac{kg}{m^3}$. The exact solution related to the problem is
$v(x,t) = \frac{1}{2}(cos(\pi(x+t)) + cos(\pi(x-t)))$. The exact solution is displayed in Figure 6. Figure 7 shows the approximated error with DNN.

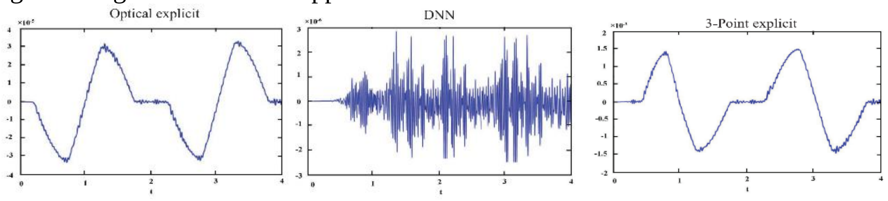

Fig. 4. Plot of the approximated error with 3-point explicit, optimal explicit, SNN and DNN

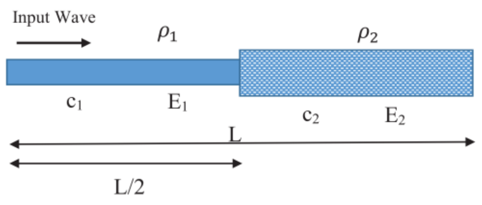

Fig. 5. Two semi-infinite strings of different densities

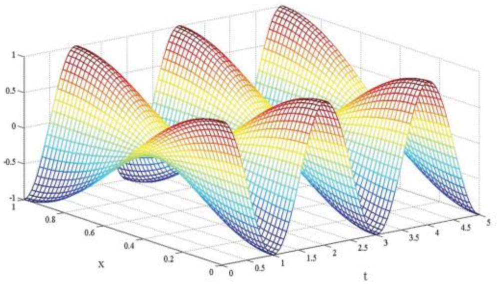

Fig. 6. Plot of the exact solution

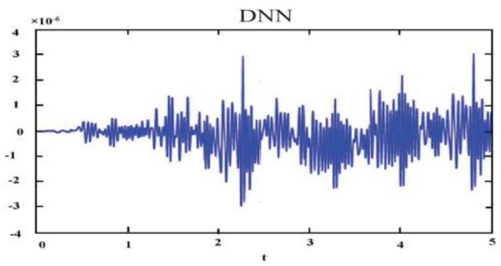

Fig. 7. Plot of the approximated error with DNN for r=70

4 Conclusions

This paper proposes a method based on the neural networks for finding the solutions of the second-order nonlinear PDEs. It is controller design process. The trial solution related to the PDE is stated as an addition of two parts. The primary part suffices the initial and boundary conditions, also does not have adjustable parameters. The secondary part includes a neural network having adjustable parameters (weights and biases). Also, a type of Bernstein neural networks namely dynamic model is proposed in order to resolve the PDEs. For obtaining the superior estimated solution of PDEs, the adjustable parameters at par with the Bernstein neural network are adjusted in suitable manner by implementing quasi-Newton learning algorithm.

Numerical examples as well as comparison with solution obtained by the employing other numerical methodologies open up that the use of neural networks based on quasi-Newton learning rule provides solutions with superior generalization and major accuracy. The future work is the application of the mentioned methodologies for system of PDEs.

References

1. Jin, L.: Homotopy perturbation method for solving partial differential equations with variable coefficients, Int. J. Contemp. Math. Sci. 3, 1395-1407 (2008).
2. Ayaz, F.: On the two-dimensional differential transform method, Appl. Math. Comput. 143, 361-374 (2003).
3. Vitanov, N.K., Dimitrova, Z.I., Kantz, H.: Modified method of simplest equation and its application to nonlinear PDEs, Appl. Math. Comput. 216, 2587-2595 (2010).
4. Wazwaz, A.M.:The variational iteration method: A reliable analytic tool for solving linear and nonlinear wave equations, Comput. Math. Appl. 54, 926-932 (2007).
5. Kharab, A., Kharab, R.: Spreadsheet solution of hyperbolic partial differential equation, IEEE Trans. Educ. 40, 103-110 (1997).
6. Kincaid, D., Cheney, W.: Numerical Analysis, Mathematics of Scientific Computing, Pacific Grove. CA: Brooks/Cole, (1991).
7. Evje, S., Karlsen, K.H.: Monotone difference approximations of BV solutions to degenerate convection-diffusion equations, SIAM. J. Numer. Anal.37, 1838-1860 (2000).
8. Bulbul, B., Sezer, M.: Taylor polynomial solution of hyperbolic type partial differential equations with constant coefficients. Int. J. Comput. Math. 88, 533-544 (2011).

9. Guo, S., Mei, L., Zhou, Y.: The compound G G ′ -expansion method and double non-traveling wave solutions of (2+1)-dimensional nonlinear partial differential equations, COMPUT. MATH. APPL. 69, 804- 816 (2015).
10. Falletta, S., Monegato, G. Scuderi, L.: A space-time BIE method for nonhomogeneous exterior wave equation problems, The Dirichlet case, IMA J. Numer. Anal. 32, 202-226, (2012).
11. Martinez, P.: A new method to obtain decay rate estimates for dissipative systems, ESAIM Control Optim. Calc. Var. 4, 419-444 (1999).
12. Gibson, J.S.: An analysis of optimal modal regulation: convergence and stability, SIAM J. Control. Optim. 19, 686-707 (1981).
13. Kroner, A., Kunisch, K.: A minimum effort optimal control problem for the wave equation, Comput. Optim. Appl. 57, 241-270 (2014).
14. Cybenko, G.: Approximation by superposition of a sigmoidal function, Math. Control Signals Systems. 2, 303-314 (1989).
15. Jafari, R., Yu, W.: Uncertainty nonlinear systems modeling with fuzzy equations, In Proceedings of the 16th IEEE International Conference on Information Reuse and Integration, 182-188, San Francisco, Calif, USA, August, (2015).
16. Jafari, R., Yu, W.: Uncertainty Nonlinear Systems Control with Fuzzy Equations, IEEE International Conference on Systems, Man, and Cybernetics, 2885-2890, (2015).
17. Jafari, R., Yu, W.: Uncertainty Nonlinear Systems Modeling with Fuzzy Equations, Mathematical problems in Engineering. 2017, (2017).
18. Jafari, R., Yu, W., Li, X.: Numerical solution of fuzzy equations with Z-numbers using neural networks, Intelligent automation and Soft Computing, 1-7, (2017).
19. Jafari, R., Yu, W., Li, X., Razvarz, S.: Numerical solution of fuzzy differential equations with Z-numbers using Bernstein neural networks, International Journal of Computational Intelligence Systems, 10, 1226-1237, (2017).
20. Razvarz, S., Jafari, R., Granmo, O.Ch., Gegov, A.: Solution of dual fuzzy equations using a new iterative method, Asian Conference on Intelligent Information and Database Systems, 245-255, (2018).
21. Dissanayake, M.W.M.G., Phan-Thien, N.: Neural-network based approximations for solving partial differential equations, Communications in Numerical Methods in Engineering, 10, 195-201 (2000).
22. He, S., Reif, K., Unbehauen, R.: Multilayer neural networks for solving a class of partial differential equations, Neural Networks. 13, 385-396(2000).
23. Montelora, C., Saloma, C.:Solving the nonlinear schrodinger equation with an unsupervised neural network: Estimation of error in solution, Opt. Commun. 222, 331-339 (2003).
24. Alli, H., Ucar, A. Y. Demir, The solutions of vibration control problems using artificial neural networks, J. Franklin Inst. 340, 307-325, (2003).
25. Sukavanam, N., Panwar, V.: Computation of boundary control of controlled heat equation using artificial neural networks, Int. Commun. Heat Mass Transfer. 30, 1137-1146 (2003).
26. Curtis, S., Ghosh, S.: A variable selection approach to monotonic regression with Bernstein polynomials, J. Appl. Stat. 38, 961-976 (2011).
27. Acosta, C.D., Burger, R., Mejia, C.E.: Monotone difference schemes stabilized by discrete mollification for strongly degenerate parabolic equations, Numer Meth Part Differ Equat. 28, 38-62 (2012).
28. Dehghan, M.: On the solution of an initial-boundary value problem that combines neumann and integral condition for the wave equation, Numer. Methods. Partial. Differ. Equ. 21, 24-40 (2005).
29. Tongue, B.H.: Principles of Vibration, Oxford University Press (2001).