International Journal of
Mathematical Modelling & Computations
Vol. 03, No. 03, 2013, 245- 252

An Application of Neural Networks to Solve Ordinary Differential Equations

S. Ezadi[a,*] and N. Parandin[b]

[a]*Department of Mathematics, Islamic Azad University, Hamedan Branch, PO. Code 65138, Iran*;
[b]*Department of Mathematics, Islamic Azad University, Kermanshah Branch, PO. Code 67189-97551, Iran.*

Abstract. In this paper, we introduce a hybrid approach based on modified neural networks and optimization teqnique to solve ordinary differential equation. Using modified neural network makes that training points should be selected over the open interval (a, b) without training the network in the range of first and end points. Therefore, the calculating volume involving computational error is reduced. In fact, the training points depending on the distance $[a, b]$ selected for training neural networks are converted to similar points in the open interval (a, b) by using a new approach, then the network is trained in these similar areas. In comparison with existing similar neural networks proposed model provides solutions with high accuracy. Numerical examples with simulation results illustrate the effectiveness of the proposed model.

Received:21 Janurary 2013, Revised:19 July 2013, Accepted:4 September 2013.

Keywords: Ordinary Differential Equations, Modified Neural Networks.

Index to information contained in this paper

1. Introduction

Differential equations are used as a powerful tool in solving many problems in various fields of human knowledge, such as physics, chemistry, mechanics, economics, etc. One application of the differential equation is turning problems and natural phenomena into differential equations, then, by solving the differential equation the

*Corresponding author. Email: Somayeh.ezadi@yahoo.com

answer is described and the phenomena are calculated. Usually many of these problems do not have analytical solutions or their solution may have certain implications. Many researchers have tried to approximate the solutions of these equations and proposed a lot of algorithms such as Runge-Kutta, Adomian, Adam-Beshforth, Adam-Molton, Predictor-Corrector methods and other methods. In recent years neural networks for estimation of ordinary differential equations (ODE) and partial differential equations (PDE) as well as the fuzzy differential equation (FDE) have been used. In year 1990 Lee and Kang [3] used parallel processor computers to solve a first order differential equation with Hopfield neural network models. Meade and Fernandez [7, 8] solved linear and nonlinear ordinary differential equations using feed forward neural networks architecture and B-splines of degree one. It is not easy to extend these techniques to multidimensional domains [1]. Lagarys and colleagues (1998) used artificial neural networks for solving ordinary differential equations (ODE s) and partial differential equations (PDE) with the initial value and boundary value problems [5]. In Comparison with jamme and Liu [4], Malek and Shekari presented numerical method based on neural network and optimization techniques which the higher-order differential equations answers approximates by finding a package form analytical of specific functions [9] which is a combination of two terms: The first term is related to the initial condition or boundary and the second term contains parameters related to the neural network, The used neural network is a two-layer network with one hidden layer and the activation function used in the hidden layer is a sigmoid function or Hyperbolic tangant function. This paper is divided into five section. In the Section 2 we introduce the basic definitions and needed theorems, In Section 3, the proposed model for solving the ordinary differential equations is introduced. In The Section 4, illustrative examples are discussed, Section 5 contains the conclusions.

2. Definitions and required theorems

This section provides the necessary Definitions and required theorems, which is used to propose the model.

Definition 2.1 (Activation functions)*In neural network, activation function is used for limiting the output neurons. In this paper we use linear activation function and Hyperbolic secont activation function. Introducing one conversion function in artificial neural network feedforward:*
heyperbolic secont transformation function

$$S(n) = \frac{2}{e^n + e^{-n}}.$$

Definition 2.2 (MLP Network training)*learning the feed forward neural networks, the law of error propagation (BP) is used which is based on The error correction learning rule. Therefore, to calculate sensitivities for the different layers of neurons in the MLP network the Derivative of conversion neurons functions is required. So functions used that have derivative. one of these functions is Hyperbolic secont function which The characteristics of this function was explained in the previous section. The Error function is described in the following sections.*

Theorem 2.3 (The World approximation Builder)*The MLP network with one hidden layer with a sigmoid functions(Hyperbolic secont function) in the middle layer and linear transformation functions in output layer are able to approximate All functions in any degree of the integral of the square. (see [2]).*

3. Problem formulation

Consider the following initial value first order differential equation:

$$\begin{cases} \frac{\mathrm{d}y(x)}{\mathrm{d}x} = f(x, y(x)), \quad x \in [a, b] \ , \\ \\ y(a) = \alpha, \end{cases} \tag{1}$$

the trial function may be written in the following form:

$$y_T(x, p) = \alpha + (x - a)N(x, p) \tag{2}$$

above function has two parts, the first part satisfies the initial condition and contains no adjustable parameters, the second part involves a neural network containig adjustable parameters [5, 9].
The error function that must minimize, has the following form:

$$E(p) = \sum_{i=1}^{m} (\frac{\mathrm{d}y_T(x_i, p)}{\mathrm{d}x} - f(x_i, y_t(x_i, p))^2 \tag{3}$$

where $\{u_i\}_{i=0}^{n-1}$ are discrete points belong to the interval$[a, b]$. Now differentiating from trial function $y_T(x_i, p)$ in (3), we obtain

$$\frac{\mathrm{d}y_T(x, p)}{\mathrm{d}x} = N(x, p) + (x - a)\frac{\mathrm{d}N(x, p)}{\mathrm{d}x} \tag{4}$$

Because derivatives of tangent hyperbolic of network feed-forward and sigmoid functions are a direct function of the value of transformation function, we do not need any differentiation. Therefore, it has many applications in hidden layer of neural network and functions such as the hyperbolic secant is less accurate in comparison with other functions. In our proposed model, modified neural network has one hidden layer with H hyperbolic secont activation functions and a linear activation function in output unit and it was shown even the use of the network by selecting a secant hyperbolic function in the hidden layer can be arbitrary accuracy than the existing neural networks for solving ordinary differential equations and fuzzy differential equations must be. w_j is a weight parameter from input layer to the ith hidden layer, v_j is an ith weight parameter from hidden layer to the output layer, b_j is an ith bias for the ith unit in the hidden layer, n_j is an output of the ith hidden unit, so in (4), $N(x, P)$ and $\frac{\mathrm{d}N(x,p)}{\mathrm{d}x}$ are defined as follow:

$$N = \sum_{j=1}^{H} V_j S(n_j) \tag{5}$$

where

$$S(n_j) = \frac{2}{e^{n_j} + e^{-n_j}}, \qquad n_j = w_j Q(x) + b_j \tag{6}$$

so that

$$Q(x) = (x+1)\varepsilon, \qquad \varepsilon \in (0,1) \tag{7}$$

therefore

$$Q(x) \in (a,b) \tag{8}$$

it means, the training points depending on the distance [a,b] are selected for training neural networks using a technique similar points in the open interval (a,b) are converted ,Then the network is trained in similar areas

$$\frac{\mathrm{d}N}{\mathrm{d}x} = \sum_{j=1}^{H} v_j w_j S'(n_j) = \sum_{j=1}^{H} v_j w_j \frac{-2(e^{w_j Q(x)+b_j} + e^{-w_j Q(x)-b_j})}{(e^{w_j Q(x)+b_j} + e^{-w_j Q(x)-b_j})^2} \tag{9}$$

using (5) and (9), (3) may rewriten the following form:

$$E(p) = -\sum_{i=1}^{m} \Big((x_i - a) \sum_{j=1}^{H} v_j w_j \frac{-2(e^{w_j Q(x_i)+b_j} + e^{-w_j Q(x_i)-b_j})}{(e^{w_j Q(x_i)+b_j} + e^{-w_j Q(x_i)-b_j})^2} - \sum_{j=1}^{H} V_j \frac{2}{e^{w_j Q(x_i)+b_j} + e^{-w_j Q(x_i)-b_j}} + f(x_i, y_t(x_i,p)) \Big)^2 \tag{10}$$

we can apply BFGS quasi-Newton method to minimize (10)[6].

4. Numerical examples

To show the behavior and properties of this new method,in this section. we discuss the simulation results one examples. The simulation is conducted on Matlab 12, the objective function in (1) minimizer engaged is fminunc. The initial weights were randomly selected.

Example 4.1 Consider the following first order ODE:

$$\begin{cases} \frac{\mathrm{d}y(x)}{\mathrm{d}x} = 4x^3 - 3x^2 + 2, \; x \in [0,1] \\ y(0) = 0 \end{cases} \tag{11}$$

The exact solution of (7) is $y(x) = x^4 - x^3 + 2x$. The trail solution is given as

$$y_T(x) = xN(x,p) = x \sum_{j=1}^{H} \frac{2}{e^{w_j Q(x)+b_j} + e^{-w_j Q(x)-b_j}}$$

thus, the error function is the following form:

$$E(p) = \sum_{i=1}^{m} \Big(\sum_{j=1}^{H} v_j \frac{2}{e^{w_j Q(x)+b_j} + e^{-w_j Q(x)-b_j}} + x_i \sum_{j=1}^{H} v_j w_j \frac{-2(e^{w_j Q(x)+b_j} + e^{-w_j Q(x)-b_j})}{(e^{w_j Q(x)+b_j} + e^{-w_j Q(x)-b_j})^2} - (4x^3 - 3x^2 + 2) \Big)^2$$

Error function in this example for the $H = 5$ hyperbolic secont units in the hidden layer and for $\varepsilon = 0.4$ $m = 6$ equally spaced points inside the interval $[0, 1]$ is trained.

the optimal value of the weights and biases are shown in Table 1 values of analytical solution and trial function are shown in Table 2. The transient behavior of proposed model in terms of the weights and biases is shown in Figure 1.

Table 1. The optimal values of weights and biases

i	1	2	3	4	5
V_i	3.0155	2.5384	-1.7355	7.5394	4.0451
W_i	2.5829	6.0664	2.1042	-4.6663	5.7714
b_i	-0.7159	2.2954	0.2143	6.4021	7.7615

Table 2. Comparison of the exact y_a and approximated y_t solutions.

i	1	2	3	4	5	6
x_i	0	0.2	0.4	0.6	0.8	1
y_t	0	0.3936	0.7616	1.1135	1.4976	1.9999
y_a	0	0.3936	0.7616	1.1136	1.4976	2.0000

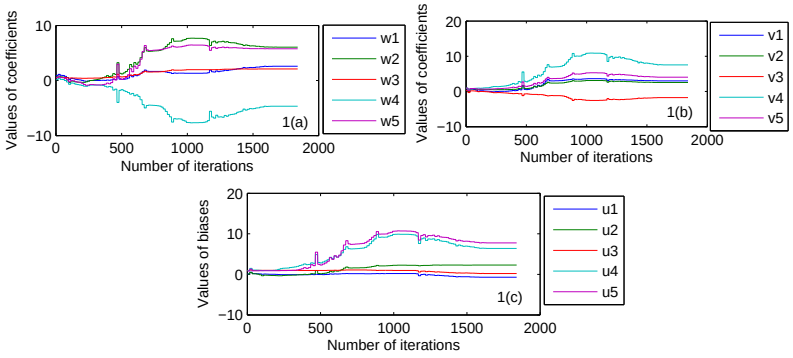

Figure 1. Source point located on the boundary, surrounded by a semicircular region.

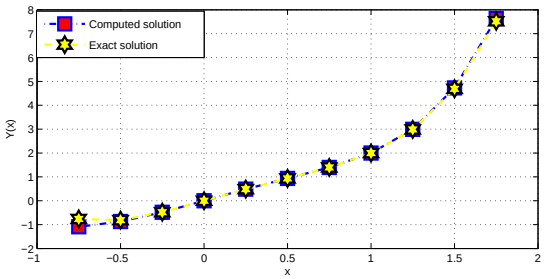

Figure 2. Numerical results for Example 1. Graph of exact solution in comparison with the computed solution inside and outside the domain $[0, 1.5]$.

The E(p) error of the network is: $2.7557e - 07$ for example 1.

Example 4.2 Consider the following first order ODE:

$$\begin{cases} \frac{dy(x)}{dx} = y(x), \, x \in [0,1] \\ y(0) = 1 \end{cases} \tag{12}$$

The exact solution of the ODE is: $y(x) = e^x$. The trail solution is given as

$$y_T(x) = 1 + xN(x,p) = 1 + x\sum_{j=1}^{H} \frac{2}{e^{w_j Q(x)+b_j} + e^{-w_j Q(x)-b_j}}$$

thus, the error function is the following form:

$$E(p) = \sum_{i=1}^{m} \Big(\sum_{j=1}^{H} v_j \frac{2}{e^{w_j Q(x)+b_j} + e^{-w_j Q(x)-b_j}} + x_i \sum_{j=1}^{H} v_j w_j \frac{-2(e^{w_j Q(x)+b_j} + e^{-w_j Q(x)-b_j})}{(e^{w_j Q(x)+b_j} + e^{-w_j Q(x)-b_j})^2} - y(x) \Big)^2$$

Error function in this example for the $H = 5$ hyperbolic secont units in the hidden layer and for $\varepsilon = 0.4$, $m = 6$ equally spaced points inside the interval [0,1] is trained.
the optimal value of the weights and biases are shown in Table 3 values of analytical solution and trial function are shown in Table 4. The transient behavior of proposed model in terms of the weights and biases is shown in Figure 3.

Table 3. The optimal values of weights and biases

i	1	2	3	4	5
V_i	-1.4517	1.7167	-0.7126	5.8092	2.0571
W_i	0.3878	0.8054	0.6515	-3.5665	-1.3531
b_i	0.8634	0.7485	1.5981	6.2970	1.8880

Table 4. Comparison of the exact y_a and approximated y_t solutions.

i	1	2	3	4	5	6
x_i	0	0.2	0.4	0.6	0.8	1
y_t	1	1.2214	1.4918	1.8221	2.2255	2.7182
y_a	1	1.2214	1.4918	1.8221	2.2255	2.7182

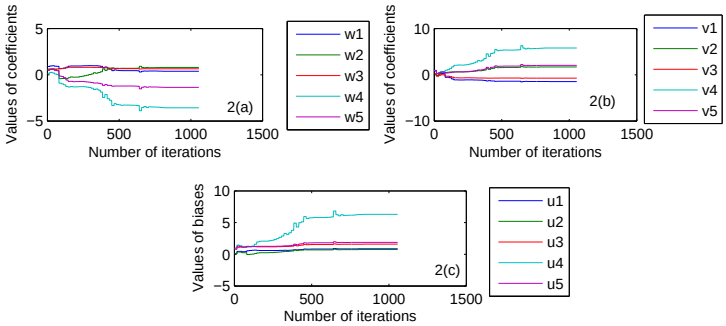

Figure 3. Transient behavior of the proposed model in Example 2: 2(a) in terms of w_1, w_2, w_3, w_4, w_5; 2(b) in terms of v_1, v_2, v_3, v_4, v_5; 2(c) in terms of b_1, b_2, b_3, b_4, b_5.

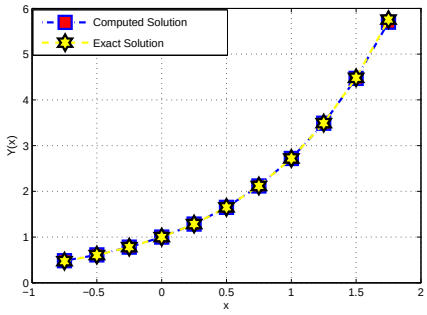

Figure 4. Numerical results for Example 2. Graph of exact solution in comparison with the computed solution inside and outside the domain $[0, 1]$.

The E(p) error of the network by B-splines of degree one [2] for example 2 is: $5.60e-05$.
and in this paper
The E(p) error of the network is: $2.3422e-07$ for example 2.

5. Conclusions

Because derivatives of tangent hyperbolic of network feed-forward and sigmoid functions are a direct function of the value of transformation function, we do not need any differentiation. Therefore, it has many applications in hidden layer of neural network and functions such as the hyperbolic secant is less accurate in comparison with other functions. In this paper, a modified neural network introduced in Section 3 was used for solving this problem and it was shown that even the use of this network by selecting a secant hyperbolic function in the hidden layer can be resulted in the arbitrary accuracy in comparison with neural networks used for solving ordinary differential equations and also fuzzy differential equations.

References

[1] Dissanayake M. W. M. G., Phan-Thien N., Neural-network-based approximations for solving partial differential equations, Communications in Numerical Methods in Engineering, **10** (1994) 195-201.
[2] Hornick K., Stinchcombe M., White Multilayer feedforward networks are universal approximators, Neural Networks, **2** (1989) 359-366.
[3] Lee H., Kang I.S., Neural algorithms for solving differential equations, journal of computational physics, **91**(1990) 110-131.
[4] Liu B., Jammes B., Solving ordinary differential equations by neural networks, in: Proceeding of 13th European Simulation Multi-Conference Modelling and Simulation: A Tool for the Next Millennium, Warsaw, Poland, June 14, (1999).
[5] Lagaris I. E., Likas A., Fotiadis D. I., Artificial neural networks for solving ordinary and partial differential equations, IEEE Transactions on Neural Networks, **9 (5)** (1998) 987-1000.
[6] Liu C., Nocedal J., On the limited memory BFGS method for large scale optimization. Mathematical Programming, **45(3)** (1989) 503-528.
[7] Meade Jr A.J., Fernandez A.A., The numerical solution of linear ordinary differential equations by feed forward neural networks, Mathematical and Computer Modelling, **19 (12)** (1994) 1-25.
[8] Meade Jr A.J., Fernandez A.A., Solution of nonlinear ordinary differential equations by feedforward neural networks, Mathematical and Computer Modelling, **20 (9)** (1994) 19-44.
[9] Malek A., Shekari R., Numerical solution for high order differential equations, using a hybrid neural networkOptimization method, Applied Mathematics and Computation, **183** (2006) 260-271.