An Intelligent Approach Based on Artificial Neural Network (ANN) to Solve Initial Value Problem

Abd El-sameea M.[1], Zaki S.[2], Abd El-wahed W.F.[3] and Ibrahim G.[4]**
1,2,4 Basic Engineering Sciences Dept – Faculty of Engineering – Menoufia Univ.
3- Faculty of Computer Sciences - Menoufia Univ.
** gamal_i_m_a@yahoo.com

Abstract

This paper presents an intelligent approach based on artificial neural networks (ANN) to obtain numerical solutions of the mathematical models of dynamical system, that are represented by ordinary differential equations (ODEs) with initial value. The intelligent approach consists of two phases. The first phase focuses on developing a simulation model of the given ODE for obtaining an approximate solution. The second phase concentrated on linking the simulation model with a feedforward ANN containing adjustable parameters (weights) . Such that the output of the first phase is used as a target of the ANN to train it. Hence we obtain ANN represent the solution of ODE. The applicability of this approach ranges from single ordinary differential equations (ODEs) , to a system of coupled ODEs. The method is illustrated by solving a variety of problems and present comparative study with the solution obtained by classical methods.

Keywords: Initial value problem; Artificial neural networks; Simulation; Backpropagation algorithm.

1. Introduction

Most problems in science and engineering are represented by a set of differential equations (DEs) through the process of mathematical modeling. Analytic solutions of the developed mathematical models based on physical laws may not be always possible. In addition, analytical methods are generally inadequate to obtain closed form solutions of the considered problems. The most used numerical methods

developed for solving DEs include finite difference method, finite element method (FEM), and boundary element method (BEM) [1-12]. Enormous progress in the field of numerical solutions of ordinary differential equations (ODEs) has been developed in the last three decades. However, these methods still based on some discretization of the domain of analysis into a number of finite elements.

Artificial neural networks is considered as an approximation scheme where the input data for the design of a network consists of only a set of unstructured discrete data points. Different strategies were developed for solving ordinary differential equations (ODEs) by using feedforward artificial neural networks (ANNs) [13–18].

Some of them produce a solution in the form of an array that contain the value of the solution at a selected group of points. Other used basis function to represent the solution in the analytic form and transform original DE to a system as algebraic equations. Many research works in solving (DEs) using ANNs are restricted to the case of solving the system of algebraic equations which result from the discretization of the domain. The solution of a linear system of equations is mapped into the architecture of Hopfield neural network. The minimization of the networks energy function provides the solution to the system of equations[19], [20], [21].

Another approach to the solution of (DEs) based on the fact that certain types of splines, for instance β_1-splines, cab be derived by superposition of piecewise linear activation function [22], [23]. The solution of differential equation with intial value using β_1 – splines as basis functions, can be obtained by solving a system of linear or nonlinear equations in order to determine the coefficient of splines. Such a solution form is mapped directly on architecture of feedforward neural network by replacing each spline with the sum of piecewise linear activation functions that correspond to the hidden unit. This method considers local basis-functions and in general requires many splines (and consequently network parameters)to yield accurate solutions.In other study [16], the numerical solutions of linear ODEs have been obtained by using multiquadratic radial basis function networks (RBFNs). The mentioned methods with different networks architectures give good results for specific class of ODEs However, none of them has produced general closed form solution. For the closed

form solution of DEs, Lagaris et al. [17] have, first, proposed and given comparisons with solutions obtained from Galerkin FEM to present performance of their method. Then, the same authors [18] have developed their method using ANN s with RBFNs.

In this paper, an intelligent methodology is developed to solve ODEs that relies on the function approximation capabilities of feedforward ANNs. The intelligent methodology consists of two phases. The first phase focuses on developing a simulation model of the given ODEs for obtaining an approximate solution. The second phase concentrated on linking the simulation model with a feedforward ANN with adjustable parameters (weights) such that the first phase is used as a target of the ANN to train it. Hence we obtain ANN represent the solution of ODE. This feedforward ANN is with one hidden layer varying the neuron number in the hidden layer according to complexity of the considered problem. The ANN has an appropriate architecture that is trained with backpropagation algorithm. This method uses feedforward neural networks with one hidden layer in which varies the neuron number as a basic approximation element, whose weights and biases are adjusted to minimize an error function. Optimization techniques requiring the calculation of the gradient of the selected error function w.r.t. the network parameter are used to train the network.

The use of simulation and ANN to solve the ODE has many attractive features such as:

1- The solution presents a very good generalization properties.
2- Less number of parameters are required than the other solution techniques. Therefore, it requires low memory space.
3- The proposed solution method is capable of applying both ODEs and PDEs.

The remainder of this paper is organized as follows: In section 2, we introduce the prposed feedforward ANN structure and backpropagation training algorithm scheme.In section 3, we proposed solutions for different example of ODEs. The obtained results are also graphically presented and some conclusive remarks are give. All computer programs developed in this paper have been performed by using MATLAB. Finally, section 4, concludes the paper.

2. The feedforward artificial neural network

A typical feedforward neural network (FFNN) consists of a number of layers of neural units. Every neural unit in a layer is interconnected to every unit in the adjacent layers.The strength of every interconnection is characterized by its weight. Information propagates from the input layer (first layer) to the output layer (last layer). Each neural unit weights the input it receives from the units in the previous layer using the appropriate interconnection weight. Subsequently, the sum of the inputs weighted is filtered through a transfer function to produce an output from the unit . The outputs from the output layer represent the final prediction of the neural network Figure (1). Training of the neural network is done by systemically adjusting the interconnection weights to minimize the error such that the predicated output from the network is as close as possible to the desired output. Several methods are used to minimize the error; for example, it is possible to use either steepest descent (i.e. the back propagation algorithm or any of its variants), or conjugate gradient method .

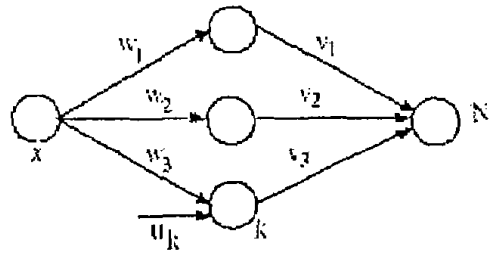

Fig.1. Neural network with one hidden layer

2.1 Learning of artificial neural networks

Learning of ANN is an algorithm for adjusting the network weights w_{ij} to minimize the difference between the actual outputs O_j and the desired output T_j

We can define an error function to quantify this difference

$$E(W_{ij}) = \frac{1}{2}\sum_{p}\sum_{j}\left(T_j - O_j\right)^2 \qquad (1)$$

This is known as the total squared error summed over all output unit j and all training patterns p.

The aim of learning is minimizing this error by adjusting the weight W_{ij}. Typically, we make a series of small adjustments to the weights $W_{ij} \rightarrow W_{ij} + \Delta W_{ij}$ until the error $E(W_{ij})$ is 'small enough'

To minimize the error function (1) we use a series of gradient decent weight updates

$$\Delta W_{kl} = -\eta \frac{\partial E(W_{ij})}{\partial W_{kl}} \qquad (2)$$

If the transfer function for the output neuron is $f(x)$, and the activations of the previous layer of neurons are z_i, then the output are $O_j = f\left(\sum_i z_i w_{ij}\right)$ and

$$\Delta W_{kl} = -\eta \frac{\partial}{\partial W_{kl}} \left[\frac{1}{2} \sum_p \sum_j \left(T_j - f\left(\sum_i z_i w_{ij} \right) \right)^2 \right] \qquad (3)$$

After repeated application of the chain rule, and some tidying up, we end up with a very simple weight update equation:

$$\Delta W_{kl} = \eta \sum_p (T_l - O_l) f'\left(\sum_n z_n w_{nl} \right) z_k \qquad (4)$$

We use a differentiable activation function such as sigmoid function in the hidden layer $\sigma(z) = \frac{1}{1+e^{-z}}$. Due to the properties of the sigmoid derivative, the general weight update in equation (4) simplifies so that it only contain neuron activations and no derivative

$$\Delta W_{kl} = \eta \sum_p (T_l - O_l) O_l (1 - O_l) z_k \qquad (5)$$

Equation (5) is known as the generalized Delta Rule for training for feedforward ANN with sigmoid activation function.

3. Solution of differential equations

In this section, we consider a solution of some problem modeled by ODEs to illustrate the efficiency of the proposed method. In all case we use a feedforward

ANN with three layers, having sigmoid activation function in the hidden layer but linear activation function in the outout layer. The ANN is trained by using backpropagation algorithms. To test accuracies of this method,the examples are also solved by either Runge-Kutta or analytical methods. Then, the obtained results are graphically presented and compared with each other. Further more,we test the method for training point and outside the training points to see approximate capability of the method for ODEs.

3.1 Solution of first-order differential equations(ODEs)

To illustrate the method, we consider the first-order OED as follows:

$$\frac{dx}{dt} = x - x^2, \quad t \in [0,1]$$
$$x(0) = 0.5$$

In the first phase we construct a simulation model of the given ODE as shown in the Figure (2).

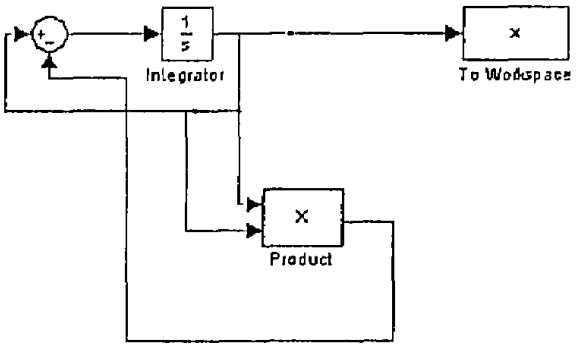

Fig. 2. ODE simulation (first phase)

The second phase is the design and train a feedforward ANN that have the following steps:

1- Initialize the ANN parameters(weights and biases).
2- Define the parameters associated with the algorthm like error goal,learning rate, maximum number of epochs(iterations),etc.
3- Call the teaching algorithm (output of the simulation phase) to train a feedforward ANN.

The network output and the solution obtained from Runge-Kutta method are shown in Figure (3). we note that the solution curves obtained from the both methods almost coincide with each other.Moreover, we can find from the network the solution points of the ODE in the outside of the training interval.

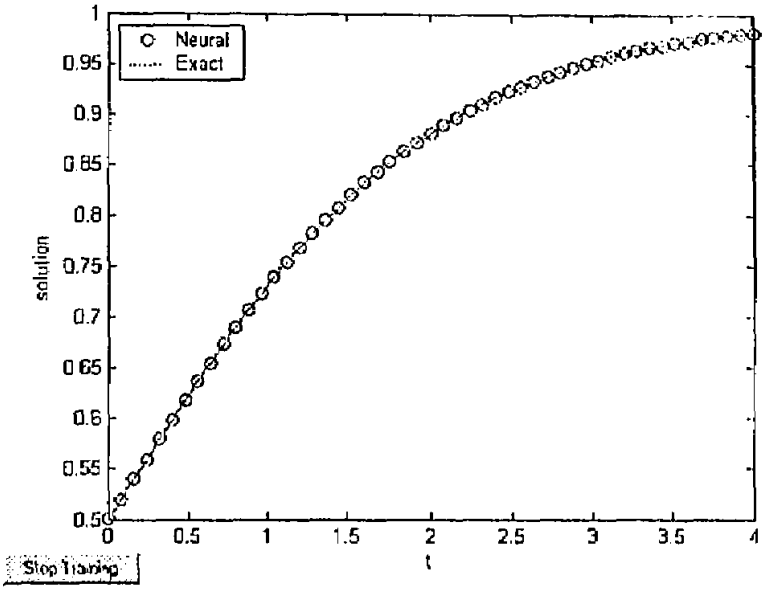

Fig. 3. Solution curves of 1st order ODE obtained with ANN and Runge-Kutta

3.2 Solution of second-order differential equations(ODEs)

In this ODE ,we consider mass-damper-spring system whose mathematical model is

$$m\frac{d^2x}{dt}+c\frac{dx}{dt}+kx=0$$

where $x(0)=1$, $\frac{dx(0)}{dt}=0$ with $t\in[0,2]$

m=c=k=1

put $x=x_1$,$\dot{x}=x_2$

Then

$$\dot{x}_1=x_2$$
$$\dot{x}_2=-x_1-x_2$$

The simulation model of this second ODE is shown in the figure (4)

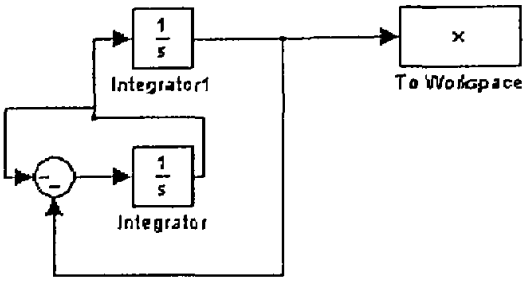

Fig. 4. Simulation of the above 2nd order ODE

Similar as 1st order ODE we design ANN to present the solution of above ODE. The network output and the solution obtained from Runge-Kutta method are shown in the Figure (5), we note that the solution curves obtained from the both methods almost coincide with each other.Moreover,we can find from the network the solution points of the ODE in the outside of the training interval

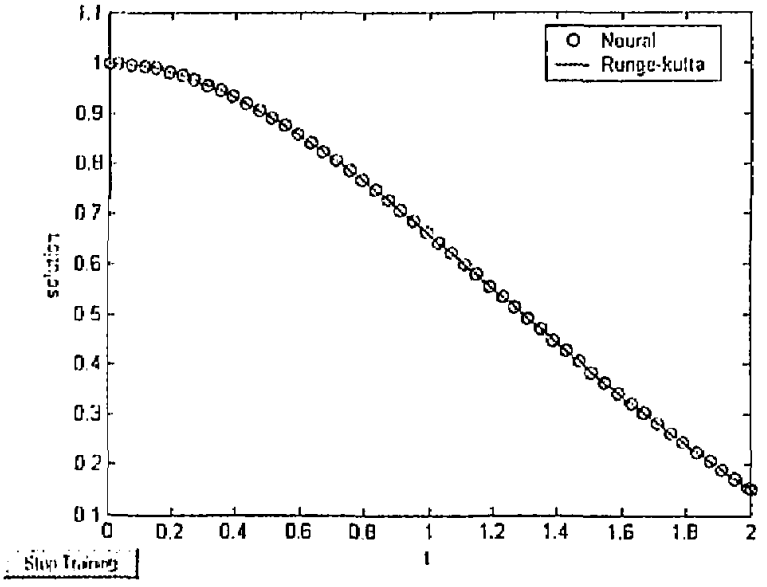

Fig. 5. Solution curves of 2nd order ODE obtained with ANN and Runge-Kutta

3.3 Solution for a system of ordinary differential equations

Let

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = x_2(1-x_1^2) - x_1$$

Where $x_1(0) = 1 \quad , x_2(0) = 1$

The simulation model of this system of ODE is shown in Figure (7)

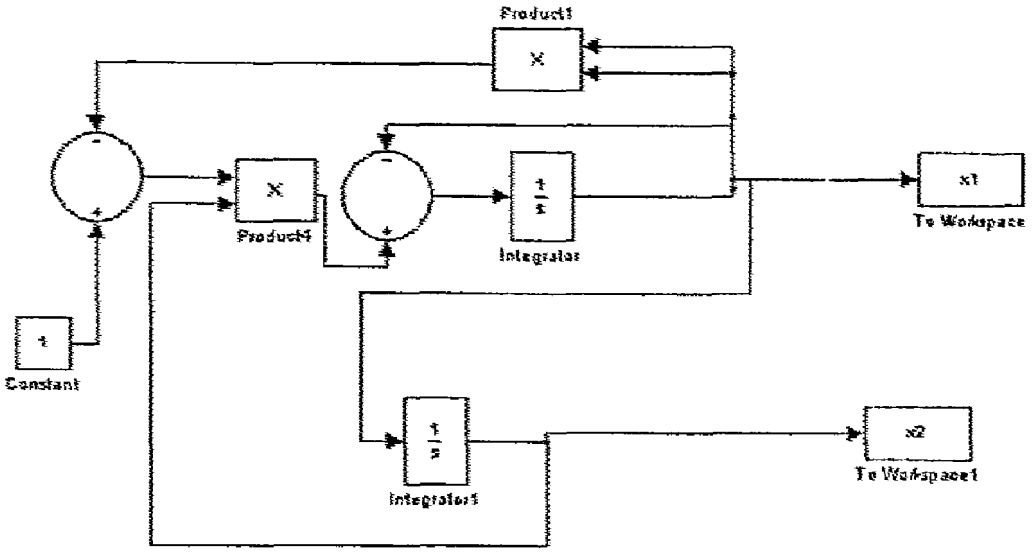

Fig. 7. Simulation of above system of ODE

The above system of ODE with two output, then the designed ANN with two neuron in the output layer. In the hidden layer we use 15 neurons to ensure the output of the ANN coincide with numerical solution of the system as shown in Figure (8).

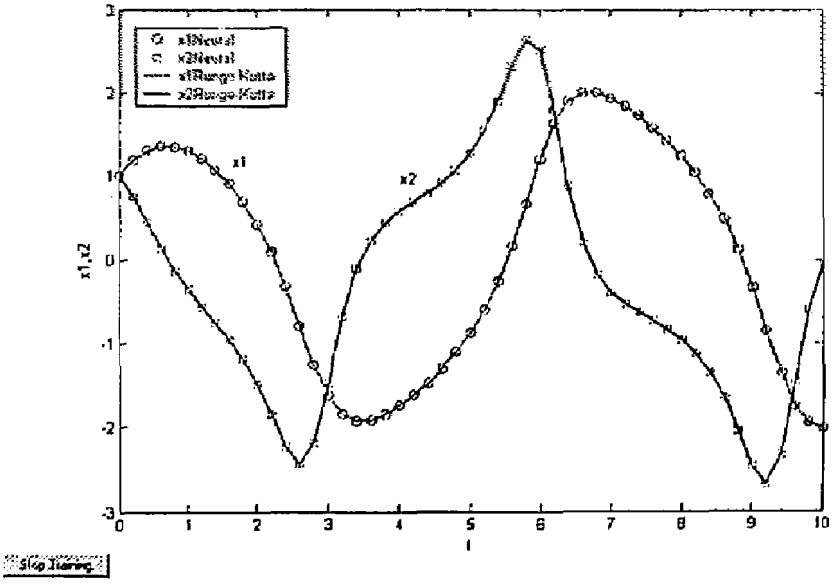

Fig. 8.Solution curves of above system of ODE obtained with ANN and Runge-Kutta

4. Conclusion

The dynamic systems are generally represented by either ODEs or PDEs. That is why we propose an alternative method using feedforward ANNs to solve ODEs. To test accuracy of this method, the problems are also solved by either Runge-Kutta or analytical methods. Then, the obtained results are graphically presented and compared with each other. Figures show that the results are in very close agreement. Further more, we test the method for training point and outside the training points to see approximate capability of the method for ODEs .

The architecture of the proposed ANN consists of one hidden layer varying its neuron number to deal with highly non-linear problems. We successfully apply this method to problems whose dynamics are represented by ODEs.

Consequently, this method can be used for wide class of linear and non-linear ODEs. Therefore, it is general and easy to apply for numerical solutions of dynamic problems.

References

[1] P.M. Lima, M.P. Carpentier, Iterative methods for a singular boundary-value problem, J. Comput. Appl. Math. 111 (1999) 173–186.

[2] H. Guoqiang, W. Jiong, K. Hayami, X. Yuesheng, Correction method and extrapolation method for singular two-point boundary value problems, J. Comput. Appl. Math. 126 (2000) 145–157.

[3] D. Kincaid, W. Cheney, Numerical Analysis, Brooks/Cole, Monterey, CA, 1991.

[4] K.S. Yee, Numerical solution of initial boundary value problems involving Maxwell's equation in isotropic media, IEEE Trans. Antennas Propagation AP-14 (1996) 302–307.

[5] P. Hunter, A. Pullan, FEM/BEM Notes, Department of Engineering Science, The University of Auckland, New Zealand, 1997.

[6] M.A. Kolbehdari, M.S. Nakhla, M.N.O. Sadiku, Hybrid model of scatteringfrom eccentrically nested dielectric cylinders, J. Franklin Inst. 225B (1999) 43–51.

[7] G. Jacobsohn, A discrete Taylor series method for the solution of two-point boundary-value problems, J. Franklin Inst. 338 (2001) 61–68.

[8] Y. Guoyou, W.J. Mansur, J.A.M. Carrer, L. Gong, Stability of Galerkin and collocation time domain boundary element methods as applied to the scalar wave equation, Comput. Struct. 74

(2000) 495–506.

[9] L. Meirovitch, T.J. Stemple, A new approach to the modeling of distributed structures for control, J. Franklin Instit. 338 (2001) 241–254.

[10] W-S. Lee, Y-H. Ko, C-C. Ji, A study of an inverse method for the estimation of impulsive heat flux, J.Franklin Inst. 337 (2000) 661–671.

[11] J. Kouatchou, Parallel implementation of a high-order implicit collocation method for the heat equation, Math. Comput. Simul. 54 (2001) 509–519.

[12] B. Bialecki, G. Fairweather, Orthogonal spline collocation methods for partial differential equations , J. Comput. Appl. Math. 128 (2001) 55–82.

[13] T. Nguyen-Thien, T. Tran-Cong, Approximation of functions and their derivatives: a neural network implementation with applications, Appl. Math. Modell. 23 (1999) 687–704.

[14] S. He, K. Reif, R. Unbehauen, Multilayer neural networks for solving a class of partial differential equations, Neural Networks 13 (2000) 385–396.

[15] C.L. Karr, I. Yakushin, K. Nicolosi, Solving inverse initial-value, boundary-value problems via genetic algorithm, Eng. Appl. Artif. Intell. 13 (2000) 625–633.

[16] N. Mai-Duy, T. Tran-Cong, Numerical solution of differential equations using multiquadric radial basis function networks, Neural Networks 14 (2001) 185–199.

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

[18] I.E. Lagaris, A. Likas, D.G. Papageorgio, Neural-network methods for boundary value problems with irregular boundaries, IEEE Trans. Neural Networks 11 (5) (2000) 1041–1049.

[19] H. Lee and I. Kang, "Neural algorithms for solving differential equations," *J. Comput. Phys.*, vol. 91, pp. 110–117, 1990.

[20] L. Wang and J. M. Mendel, "Structured trainable networks for matrix algebra," *IEEE Int. Joint Conf. Neural Networks*, vol. 2, pp. 125–128,1990.

[21] R. Yentis and M. E. Zaghoul, "VLSI implementation of locally connected neural network for solving partial differential equations," *IEEE Trans. Circuits Syst. I*, vol. 43, no. 8, pp. 687–690, 1996.

[22] A. J. Meade, Jr., and A. A. Fernandez, "The numerical solution of linear ordinary differential equations by feedforward neural networks," *Math. Comput. Modeling*, vol. 19, no. 12, pp. 1–25, 1994.

[23] , "Solution of nonlinear ordinary differential equations by feedforward neural networks," *Math. Comput. Modeling*, vol. 20, no. 9, pp.19–44, 1994.

[24] J.W. Hines, Fuzzy and Neural Approaches in Engineering MATLAB Supplement, Prentice-Hall, Englewood Cliffs, NJ, 1997.